

UNIVERSITY OF CANTERBURY

MASTERS THESIS

**Performance and Security Analysis
of Flexible Random Virtual
Internet Protocol Multiplexing on a
Virtualised Network**

Author:
Cole DISHINGTON

Supervisor:
Dr. Dong Seong KIM

*A thesis submitted in fulfilment of the requirements
for the degree of Masters of Computer Science*

in

Computer Science and Software Engineering

February 5, 2019

UNIVERSITY OF CANTERBURY

Abstract

College of Science

Computer Science and Software Engineering

Masters of Computer Science

Performance and Security Analysis of Flexible Random Virtual Internet Protocol Multiplexing on a Virtualised Network

by Cole DISHINGTON

With the interconnection of services and customers, network attacks are now capable of large amounts of damage. Defending against these attacks is difficult, especially with an asymmetric disadvantage between defender and attacker. Flexible Random Virtual internet protocol Multiplexing (FRVM) is a moving target defence technique that protects against reconnaissance and access with address mutation and multiplexing. By deploying FRVM on a Mininet network, this thesis demonstrated its security and performance trade-off. Address mutation and multiplexing can largely increase the duration of network scans and obfuscate the results. Additionally, address multiplexing is inexpensive in terms of performance. These could drive future research to include address mutation and multiplexing. Finally, the methodology and results could fuel future research into FRVM.

Acknowledgements

Firstly, I would like to thank my thesis supervisor, Dr. Dong Seong Kim of the University of Canterbury, for his professional advice and comments throughout my masters thesis. With his invaluable advice and comments, I have been steered in the right direction when lost. Additionally, I appreciate the funding from the US Army Research Lab towards development and evaluation of software defined networking based moving target defences, only made possible by Dr. Dong Seong Kim.

I would also like to thank Dilli Sharma, a co-inventor of Flexible Random Virtual internet protocol Multiplexing (FRVM), for his help in understanding FRVM. This aided in creating a software implementation that was representative of the theoretical technique.

I would also like to thank my supportive partner, Kate Olsen, for proofreading my thesis time and time again. This thesis would not be to its current quality without her help.

Finally, I would like to thank the University of Canterbury's statistical consultants, Dr. Daniel Gerhard and Dr. Elena Moltchanova, for their time and help.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation for Study	2
1.2 Aim and Scope	4
1.3 Research Questions	5
1.4 Approach and Outcomes	5
1.5 Organisation of the Thesis	6
2 Literature Review	9
2.1 Issues with MTD Integration	10
2.2 Network-Based MTD Techniques	11
2.3 Assessment of Network-Based MTD Techniques	21
2.4 Summary	28
3 Method	31
3.1 Threat Model	32
3.2 Overview of FRVM	33
3.3 Metrics of Evaluation	40
3.4 System Software	42
3.5 Network Configuration	43
3.6 SDN Controller Implementations	47

3.7 Measurement Instruments	57
3.8 Data Collection	59
3.9 Data Analysis	64
3.10 Reproducibility of the Experiment	66
3.11 Summary	66
4 Results and Analysis	69
4.1 Security Analysis	69
4.2 Performance Analysis	90
4.3 Discussion	98
4.4 Limitations of the Study	102
5 Conclusion	107
References	111

List of Figures

3.1	Communication between two FRVM protected hosts in the same subnet.	34
3.2	Difference between RSM and FRVM with respect to information disclosed by a port probe.	37
3.3	The basic topology of the virtualised network.	44
3.4	The spread of an ARP broadcast in a FRVM deployed network.	48
3.5	Switches flow tables in a FRVM deployed network.	49
3.6	Switches flow tables in a tSDN deployed network.	53
3.7	Switches flow tables in a RSM deployed network.	54
4.1	Sample distributions of the number of discovered hosts whilst scanning tSDN and RSM deployed networks, for full scans with multiple scan types.	72
4.2	Sample distributions of the average number of discovered open ports whilst scanning tSDN and RSM deployed networks, for full scans with multiple scan types.	73
4.3	Sample distributions of the duration of scans on tSDN and RSM deployed networks, for full scans with multiple scan types.	74
4.4	Sample distributions of the number of discovered hosts whilst scanning RSM and FRVM deployed networks, for partial scans with multiple different scan types.	77

4.5	Sample distributions of the average number of discovered open ports whilst scanning RSM and FRVM deployed networks, for partial scans with multiple different scan types.	78
4.6	Sample distribution of the number of discovered hosts whilst scanning a RSM deployed network, with a full TCP connect scan for multiple mutation intervals.	83
4.7	Sample distribution of the average number of discovered open ports whilst scanning a RSM deployed network, with a full TCP connect scan for multiple mutation intervals.	84
4.8	Sample distribution of the duration of scans on a RSM deployed network, with a full TCP connect scan for multiple different mutation intervals.	85
4.9	Sample distributions of the number of discovered hosts whilst scanning RSM and FRVM deployed networks, with a partial TCP connect scan for multiple mutation intervals.	87
4.10	Sample distributions of the average number of discovered open ports whilst scanning RSM and FRVM deployed networks, with a partial TCP connect scan for multiple mutation intervals. . .	88
4.11	Line plot of TCP file transfers against the number of concurrent connections to different protected servers.	92
4.12	Line plot of the duration of TCP file transfers against the number of concurrent connections to a single protected server. . . .	93

List of Tables

3.1 Software specifics of the development environment for the thesis.	44
4.1 Two full scans collected from a FRVM deployed network. . . .	72
4.2 Summary statistics for the explanatory variables of an AFT model fitted to the duration of TCP file transfer.	91
4.3 Hypothesis testing for the duration of TCP file transfers with different SDN controllers.	91

List of Algorithms

1	FRVM controller's vIP address generation.	49
2	FRVM controller.	50
3	FRVM's multiplexing and demultiplexing with ARP, a portless protocol.	52
4	tSDN controller.	52
5	RSM controller.	55
6	RSM controller's vIP generation.	55

Dedicated to my partner Kate

Chapter 1

Introduction

As the digital world has developed and grown, it has increasingly become integrated into our society and lives. Though this progression important data and services, such as personal data, trade secrets, and financial and emergency services, have been migrated to the Internet [3]. During this time, the criminals that sought to wreak havoc and gain personal wealth outside of the digital world have followed suit. In the digital world, these criminals have been labelled attackers, or adversaries. Network security plays a vital role in the digital world to protect the confidentiality, integrity, and availability of data and services against the threats posed by attackers [45]. A magnitude of past and ongoing research has been conducted to fortify networks, attempting to make them impenetrable to attackers; however, attackers continue to either break down or squeeze through a hidden hole in built-up security [46]. This is because the attackers have an asymmetric advantage against defenders [49]; the defences remain static, but the threats can morph as an attacker learns more about the unchanging defences and its weaknesses. Moving Target Defence (MTD) was designed to level the playing field by modifying the attack surface proactively, reactively, or a combination of each to make studying defences difficult and time sensitive. With MTD, an attacker no longer

has copious amounts of time to study and familiarise themselves with defences. With the surge of interest in security, especially in MTD, there have been many MTD techniques proposed in recent years [30] although most only exist in academia without complete analysis including security and performance evaluation. This includes Flexible Random Virtual internet protocol Multiplexing (FRVM) [42], a recent MTD shuffle technique proposed to proactively mutate protected hosts Internet Protocol (IP) addresses per service to invalidate attackers network scanning attempts. Currently, FRVM is theoretical with analytical security analysis. Before FRVM can be trusted for deployment, it must be evaluated in terms of performance and security through realistic evaluation to ensure compatibility with existing network infrastructure. Without thoroughly evaluated MTD available, the techniques will not be adopted, and the asymmetric advantage of the attacker will remain. Unfortunately, methods for analysis of MTD are still under debate and most methods use analytical models [3], that do not capture real-world considerations for deployment.

1.1 Motivation for Study

This thesis will address three challenges present in MTD security: real-world considerations for deployment, analysis of MTD, and evaluation of FRVM. With the surge of interest in MTD, many techniques have been proposed although often these techniques avoid discussion of potential problems with common network infrastructure and susceptibility to real-world attacker tools. These problems can be addressed through implementation and deployment alongside common network infrastructure, and realistic evaluation of security and performance. Unfortunately, realistic evaluation has been lacking in

similar techniques to FRVM. Critique of past MTD techniques in the literature review will aid in future research. Additionally, past and future MTD techniques, especially address masking techniques, could make use of the realistic evaluation described in the methodology. Surprisingly, these considerations are missing in many address masking MTD and collected results often lack statistical rigour.

FRVM is a promising recent addition to MTD, introduced to improve upon the existing address masking techniques [49, 41, 23, 22, 21] although it is currently theoretical with only analytical security analysis. With a virtualised network, this thesis will realistically evaluate FRVM, measuring its security and performance. FRVM will benefit in both short and long-term from implementation, deployment on a network, evaluation through detection of design issues, practical analysis for support of the analytical results, and additional future research directions. The evaluation will provide extra insight into the technique through testing FRVM against network scanning software used by attackers in real-world attacks [28]. Trust in the technique will also increase as the results will be reproducible on other systems. Lastly, the evaluation of FRVM will benefit the field of security through evidence of FRVM's ability to defend networks from attackers' scans and understanding of the performance costs. Increased trust in MTD and FRVM will follow, aiding the adoption of MTD to real networks. This would abolish the asymmetric advantage of the attacker against the defender, forcing attackers to design more complicated MTD aware reconnaissance tools. The research outcomes of the thesis were met in the allocated time through assumptions in the research design: the attacker was not aware of the MTD, attacker had to scan before launching an attack, and a virtualised network accurately models the delays of a physical network.

1.2 Aim and Scope

The aim of this thesis is to realistically evaluate FRVM through deployment on a virtualised network and evaluate FRVM with respect to security and performance. The scope will involve a review of literature, implementation and design of the experiment, and collection and analysis of security and performance metrics. Literature regarding issues with MTD integration, network-based MTD techniques, and the assessment of network-based MTD techniques will be reviewed. The virtualised network will be configured before implementing the controllers. Therefore, the controllers can be debugged on the network throughout their development. After implementation of controllers, design of the tests will follow to collect metrics. Upon collection of the metrics, analysis will be conducted to draw conclusions from the experiment.

The limitations of this approach include using a virtualised network, assumption of attack chain, and attacker's unawareness of MTD. Because of the virtualised network, the exact runtime of the controller or switches fitted with specialised hardware is unknown. Although the virtualised controller and switches operate software used in real-world SDN deployments, therefore the trends should approximate the true runtime. The security analysis assumes an attack chain in which reconnaissance and access are the first two steps although this is not always the case. An example to the contrary is CryptoLocker [24], that spread through social engineering via email. Lastly, security analysis with Nmap [28] assumes an attacker that is not MTD aware. Potentially, modifications to network scanning software could improve attackers' ability to scan FRVM protected networks.

1.3 Research Questions

The purpose of this thesis is framed around the following research questions:

- What extra considerations will need to be handled to implement and deploy FRVM?
- How does FRVM affect security scanning in terms of information disclosure and scan duration?
- What is the effect of FRVM on a network's performance?

1.4 Approach and Outcomes

This thesis involved implementing and deploying FRVM on a virtualised network for security and performance analysis. FRVM was implemented with a Network Operating System (NOS), in addition to two comparison SDN controllers. The comparison controllers implemented Ethernet-level routing and IP-level routing with address mutation, respectively. Each controller was deployed on the virtualised network and subjugated to security and performance analysis.

Security was measured with a real-world network scanning tool, Nmap. Different port scanning techniques were tested, with the most threatening tested with respect to multiple mutation intervals. The level of security was measured with quantitative metrics for information disclosure and scan duration. In turn, the metrics were analysed with exploratory statistics.

Performance testing involved measuring the duration of TCP file transfers for different levels of network load. This produced quantitative metrics for the duration at different loads, these were analysed using survival analysis.

This thesis discovered and solved issues with FRVM and existing network infrastructure. The discovered issues were with Address Resolution Protocol (ARP) and default route compatibilities. Additionally, the thesis analysed security and performance of controllers. The security benefit of address multiplexing was found to be much greater than its performance cost. Whereas, the performance cost of address mutation was significant. Although, the benefits of address mutation at a high frequency was found to hugely increase attackers network scan duration, as well as obfuscating and hiding protected network information.

1.5 Organisation of the Thesis

The following outlines the structure of the thesis. Chapter 2 describes relevant literature, Chapter 3 the methodology of the thesis, Chapter 4 the results and analysis, and Chapter 5 the discussion and conclusion. The literature review summarises contributions to the issues with MTD integration, network-based MTD techniques, and the assessment of network-based MTD techniques. The methodology details the experiment setup including necessary software and network creation, FRVM coupled with resolved ambiguities and the software implementation of FRVM and comparison techniques, measurement instruments and data collection, and finally the statistical methods employed for data analysis. The results analyse collected security metrics with exploratory statistics and performance metrics with survival statistics, leading to considering the contributions and implications of the thesis. The

conclusion summarises the thesis with respect to the main points drawn in each chapter.

Chapter 2

Literature Review

The literature review will address three topics in Moving Target Defence (MTD): integration issues, network-based MTD techniques, and assessment of network-based MTD techniques. In the past decade, MTD has gained a lot of interest in security research due to its ability to reduce the defenders' asymmetric disadvantage against attackers, leading to the proposal of many MTD techniques. Although many new techniques have been formulated almost none have found wide deployment, due to the uncertainty surrounding MTD performance costs and security benefits [30]. Further, network-based MTD can invalidate both host and network processes. Before more MTD techniques can be deployed into real networks their defence must be proven through both analytical and realistic evaluation. The evaluation should involve interaction with real-world protocols, aiding in finding issues with existing infrastructure. The following sections introduce concepts related to issues with MTD integration, past MTD techniques including many that defend against reconnaissance, and assessment performed on past MTD techniques.

2.1 Issues with MTD Integration

Security techniques are designed to protect systems and networks against attackers. This can involve blocking network traffic and applications or denying permission to resources. While these provide protections, it is important not to risk the reliability of the protected systems or networks. MTD revolutionises security by removing the asymmetric disadvantage of the defender against the attacker, although MTD come with extra reliability issues that must be overcome. MTD introduce these issues through the dynamics that are applied for protection. Luckily, in the past decade many MTD techniques have been devised in the excitement of crushing the asymmetric disadvantage; these have been surveyed and design methodologies have been born, helping new MTD to avoid impairing the reliability of protected systems or networks.

The dynamics of MTD techniques can cause issues with existing infrastructure and violate assumptions that are valid for traditional networks, hindering the legitimate use of protected systems. For network-based MTD these include mutation on the network topology [18], network addresses [29], and service port numbers [22, 31]. Due to the randomisation of network properties [30] implicit assumptions of protocols and common infrastructure can be violated. An example is the requirement of a web server to be in a known location, which could be violated by the randomisation of network addresses. Further, even if the network properties are not applied to a web server but instead to the client, the dynamics can break the client-server model. It follows that the issues can be broad although all these issues are not present in every network-based MTD technique, neither is the severity the same between techniques. For this reason, a MTD technique must be thoroughly analysed to ensure issues with existing infrastructure do not remain undiscovered.

In the analysis of MTD techniques there is a focus on analytical analysis of MTD techniques [3]. This is because the creators want a universal tool, that can readily be applied to any MTD technique. While these tools are useful in the design process of a technique [14], they tend to ignore specific details in a MTD technique [51, 55], avoiding a realistic evaluation. Inherently, realistic evaluation focuses on the technique, losing the ability of universal application. Although focusing on the technique allows the evaluation to discover and solve issues specific to the technique. Therefore, without realistic evaluation of MTD techniques, integration issues with existing infrastructure can be difficult to discover.

2.2 Network-Based MTD Techniques

Traditional network and system configurations are static, within a large period, and therefore usually remain the same after detection by network scanning. Fortunately, dynamics can be applied to a network or system with MTD to frustrate attackers with extra complexity and time investment. MTD techniques aim to frustrate the attacker at some stage in the attack chain [30]. FRVM, the focus of this thesis, aims to stop attacker reconnaissance and access, the first two links of the attack chain. By protecting links at the beginning of the attack chain, FRVM can stop attacks before they start and even protect against zero-day attacks. For these reasons, MTD that protect these links are of particular interest. The following describes and evaluates network-based MTD techniques that defend against one or both of these links. Evaluation of the analysis conducted on MTD techniques is withheld until Section 2.3.

2.2.1 Address Mutation Against Reconnaissance

Address mutation techniques involve, usually proactive, randomisation of an identifying address, typically an IP address. Coordinating, generating, and allocating these addresses incurs extra overhead. Two common approaches regarding host visibility are transparent and non-transparent. Transparent approaches hide the address randomisation from the protected hosts, avoiding any additional overhead or configuration on the hosts. Whereas, non-transparent approaches involve the hosts to utilise existing hardware and share the load.

Transparent Address Mutation Against Reconnaissance

Transparent address mutation techniques tend to achieve their transparency from hosts by introducing a virtual address space. The resultant network then has two address spaces: real and virtual. Real addresses are known by the hosts and the MTD technique. Virtual addresses are known by outside users communicating with protected hosts and the MTD technique. The MTD technique is aware of both, this allows it to map between the addresses without the knowledge of hosts. The benefit of including the virtual address space is that upon address mutation, nothing has changed for the protected hosts since they do not know their virtual addresses.

Over the years, a research group has progressively built upon their address mutation technique. Beginning with OpenFlow Random Host Mutation (OF-RHM) [23], the technique sought to mask end hosts IP addresses by introducing a virtual IP (vIP) address space managed with Software Defined Networking (SDN). Each host in an OF-RHM network is allocated a real IP (rIP) address and vIP address. The vIP address is mutated continually after

some interval chosen individually for each host based on a decided security level. Communication with protected hosts is through vIP addresses that are mapped by a SDN controller to the corresponding rIP address. This allows OF-RHM to function transparently to end hosts, with only the SDN controller and switches aware of the vIP address space. Transparency is achieved by switches on the edge of the network mapping between the vIP and rIP address spaces. Chunks of the vIP address space are allocated to each subnet in an OF-RHM network, in which allocation of vIP addresses to end hosts is performed either with a uniform probability or with some predefined weighting. To increase the unpredictability of the allocated vIP addresses, the interval in which no vIP addresses are allocated to the same host twice is maximised for each subnet using a variation of the knapsack problem.

Moving to Random Host Mutation (RHM), the technique was adjusted to function on traditional networks without SDN. Additionally, address mutation stages and Transmission Control Protocol (TCP) session tracking were added on top of OF-RHM. The address mutation stages were introduced to optimise, in terms of time, the allocation of many vIP addresses to hosts. The stages are Low frequency Mutation (LFM) and High Frequency Mutation (HFM), and many HFMs occur in a single LFM. LFM allocates a range of virtual IP addresses (VAR) to each host, ensuring to minimise the size of unused address space. For each host, HFM randomly selects a single vIP address from the set of VAR, allocated by the LFM stage. Thus, LFM must ensure each host has a VAR which relates to the hosts' specific mutation rate. The LFM and HFM were added to speed up mutation rate and hence improve unpredictability. RHM performs TCP session tracking to ensure connectivity between end hosts is not lost between address mutations. Thus, upon a HFM on a host, any new incoming sessions receive the new vIP address although all old sessions will communicate with their original vIP address. Therefore,

a host may be associated with multiple vIP addresses at any one time.

Finally, retaining the name RHM [22, 21] the technique employs short-lived ephemeral IP (eIP) addresses that are like the vIP addresses that were introduced in both OF-RHM [41] and RHM [41]. Although RHM now randomises Media Access Control (MAC) addresses and domain names and further builds upon the address allocation by detecting scanning to modify the allocation of eIP addresses. Thus, the eIP addresses are not randomly selected uniformly, like other address randomisation techniques, providing an adversary-aware proactive defence. This further ensures that a naive attacker cannot scan the eIP address space.

Downsides of these techniques include tracking connections, retention of vIP addresses, static access information, higher demand of IP addresses, and possible address collisions. The techniques track TCP connections to ensure addresses associated with concurrent connections are retained. Due to the vast number of connections a server can have this can become expensive. Additionally, the retained vIP addresses can lead to exhaustion of the IP address space. This would be easily exploitable by an attacker using a Slowloris attack [52]. Despite the movement of addresses, any data collected in the access stage can be stored and used to increase the speed of scanning. For example, port information could be used for rediscovery or fingerprinting of a specific host. With the addition of the LFM and HFM mutation stages, more unused addresses are needed to avoid address collisions. Unfortunately, IP addresses are a constrained resource. Further, the address pool, VAR, is generated from addresses that are not in ongoing connections. It follows that a smaller scale Slowloris attack could significantly increase the chance of address collisions. With the generation of VARs, further address collisions are possible. The vIP addresses in VARs are generated with hash functions, the resultant hash is feed through modulo arithmetic to ensure the IP address fits

in the specified subnetwork. Typically, hash functions have high collision resistance although modulo arithmetic will reduce the variability of the hashes. This will increase the chance of an address collision within a VAR.

Non-Transparent Address Mutation Against Reconnaissance

Non-transparent address mutation techniques include protected hosts in the technique, distributing the workload. Although, including hosts requires modification of their systems.

Since the introduction of MTD, there have been many techniques that attempt to limit attackers' ability to perform network reconnaissance. Although this paper [29] suggested that past techniques either have scalability issues or cannot communicate with unmodified clients. SDN shuffle was proposed to improve upon past address masking techniques by shifting load onto the protected network hosts and compatibility with TCP. The load is shifted by the SDN controller intercepting DNS replies and instructing hosts of the Network Address Translation (NAT) rules that should be applied to the packets. This avoids the SDN controller and switches sustaining all the extra overhead. Although, without transparency from the hosts, they become more complex, requiring additional software. Compatibility with TCP is achieved by allocating a new address per connection. Although this implementation is simple, it could lead to address exhaustion quickly. Additionally, for the SDN controller to generate IP addresses for each connection the TCP sessions need to be tracked. This could lead to a large overhead with many connections. Lastly, the NAT rules are installed on the client and server although hosts tend to lack any specialised networking hardware.

Network Address Space Randomisation (NASR) [1] was an early attempt at limiting the spread of hitlist worms with address mutation. Hitlist worms

pose a greater risk than usual self-propagating worms due to their increased speed. Assembling a list of vulnerable machines in advance of the attack, hitlist worms can avoid host discovery during their spread. NASR aims to combat their ability by mutating addresses over a long interval, slowly expiring the attacker stored information. Intervals of newer address mutation techniques change within minutes whereas NASR changed within hours. Formally, NASR is a proactive MTD that mutates addresses with the use of Dynamic Host Configuration Protocol (DHCP). Protected hosts are reminded to mutate by NASR, although if the host is in an ongoing connection NASR will not send the reminder until a limit of deferrals has been reached. This approach would likely incur a lower performance penalty than newer address mutation technique due to the slower address mutation. Although, this approach still tracks ongoing TCP connections. Disadvantages include the security and performance trade-off and slow mutation rate.

2.2.2 MT6D Based techniques

As more network infrastructure transitions over to Internet Protocol version 6 (IPv6), some address masking techniques have adopted IPv6. Due to the formation of IPv6, privacy-related crimes are easier for an attacker due to extra address assignment mechanisms that were not available in IP version 4. The Moving Target IPv6 Defence (MT6D) [9, 10, 38] aimed to solve privacy concerns introduced in IPv6 in addition to disrupting eavesdropping, replay attacks, and other attacks that assume a static address. As a proof of concept, privacy concerns were demonstrated by setting up a Wireless Sensor Network (WSN) over Virginia Tech's campus to track android mobile phone that travelled between networks on campus. MT6D mutates network addresses

with high frequency, however, a modified network address will disrupt existing connections. Thus, MT6D tunnels packets inside of a User Datagram Protocol (UDP) packet, and both communicating clients must run MT6D to encapsulate and decapsulate outgoing and incoming packets, respectively. During the process of encapsulation, the source and destination addresses are stripped from the packet and the encapsulating UDP packet has source and destination set to the current random addresses. Upon arrival, the data is decapsulated and the original source and destination addresses are replaced. To achieve this, hosts must share a common secret before communication to generate the random addresses. MT6D can be run both transparently and non-transparently. Transparency can be achieved by running the encapsulation and decapsulation on a gateway, or it could run non-transparently on the host machine. MT6D has been proposed as a protection for the smart grid [15], given that it is running transparently. Weaknesses of MT6D include that either a complex gateway or increased complexity of hosts is required, all communicating parties must have a MT6D setup, tight time synchronisation is needed to ensure mutated addresses are consistent between communicating parties, and there are no address allocation considerations in the network thus there may be collisions.

With the rise of the Internet of Things (IoT) over the past years [6] many different products have been released that collect personal data with the aim of personalising users' experience. Collected personal data must be protected introducing challenges with past methods due to mobile and resource constraints inherent in IoT devices. μ MT6D [54, 53] is an adaptation of MT6D [9, 38] to function on IoT devices. To achieve deployment on IoT, optimisations had to be made including converting source code from Python [36] to C, designing a more efficient dynamic address change, and using lightweight hash algorithms. Currently, the last two points have not been implemented

into μ MT6D although the current implementation has been deployed onto two different testbeds: simulation software and a small number of WSN nodes. Weaknesses of μ MT6D include those already existing in MT6D in addition to the requirement of a μ MT6D gateway for limited IoT and the current progress of the project. Additionally, most IoT are limited and many are mobile thus the requirement for a μ MT6D gateway for Internet connectivity could be frustrating unless deployment was vast.

Users wishing to make their connection secure and private usually opt for a Virtual Private Network (VPN). Although the security and privacy of VPN users rely heavily on the entry and exit points remaining undiscovered. The Moving Target Mobile IPv6 Defence (MTM6D) [17] was designed to apply MT6D concepts [9] to a VPN and additionally improve upon shortcomings of MT6D. The improvements include the possibility of address collisions and the need for time synchronisation between gateways. In a follow-up paper, the application of MTM6D to VPN servers was suggested as an anti-censorship platform [16]. The main difference between MTM6D and MT6D is the use of mobile IPv6 addresses and the associated mechanisms built into IPv6 to generate IP addresses. This generates a single static Home Address (HoA) and many temporary Care-of-Addresses (CoAs) that act as the vIP addresses. A CoA is generated for every host connecting to the VPN, and due to the enormous address space of IPv6 and the number of connections a VPN server can handle this should be achievable. Although a network scan has the possibility of finding the HoA, thus bypassing the anonymity afforded by the CoAs.

2.2.3 Address Mutation Against Reconnaissance and Access

There have been many IP address related MTD techniques released although privacy issues still exist. Flexible Random Virtual IP Multiplexing (FRVM) [42] was proposed as an alternative to past IP address mutation techniques with the addition of service multiplexing per host to protect against access. Similar to OF-RHM [23] and RHM [41, 22, 21], vIP addresses are mapped from hosts rIP addresses to vIP addresses by switches at network edge. The vIP address space is mutated on a constant interval although at this point there is no method of optimising the distribution of vIP, as in RHM [41]. FRVM is deployed on a controller in a SDN-based network to install necessary flows on switches for address mutation. The novel addition that FRVM introduces beyond past address masking techniques is that rather than allocating vIP addresses per host, FRVM allocates vIP addresses per service. Therefore, a single host can have multiple vIP addresses, further improving defence against reconnaissance and access and lowering the asymmetric advantage of the attacker. Disadvantages include incompatibility with the TCP protocol and simple address allocation. The incompatibility comes from the vIP addresses mutating, this violates TCP's assumption that IP addresses will remain the same over the duration of connections. The address allocation is simple and has not been improved to increase efficiency.

2.2.4 Obfuscation of Network Information

Obfuscation of network information involves invalidating information in packets that could be useful to attackers. Usually, this is achieved with the help of an Intrusion Detection System (IDS) to detect potential attackers. Detection allows the obfuscation to only be applied to malicious packets. However,

detection is never perfect and false positives are always going to occur. The following techniques obfuscate network information to confuse attackers.

The aim of CHAOS [43] is to invalidate network reconnaissance and access information gained by the attacker through obfuscation of network configurations. CHAOS aims to achieve this through detection and obfuscation of unexpected traffic in terms of IP address, port numbers, and system fingerprint. In addition, CHAOS aimed to streamline this process to enable deployment on real networks. To aid in this, a data structure called a CHAOS Tower Structure (CTS) was developed to optimise the detection of unexpected traffic. The CTS is essentially a threat pyramid in which hosts are placed in descending level of risk, evaluated based on ease of exploitation and impact if exploited. Connections are flagged if initiated from a higher layer to a lower layer and are then judged by an IDS, unless the connection is marked as special. Special connections are those connections that are expected between the two hosts, regardless of the hosts' placement in the CTS. These are usually connections which are important to the functionality of the network. Weaknesses of CHAOS include attacks spreading to lower layers and static detection of traffic. If an attack has occurred on a higher layer of the CTS then CHAOS makes no effort to stop the spread to lower layers. CHAOS does not obfuscate all connections and thus relies on detection by an IDS, although an IDS is only as accurate as the signatures it is configured to detect. It follows that an attacker using an unknown scanning technique or stealthy scan would not have their results obfuscated as there will not be any signatures able to match the scanning technique.

Traditional protection mechanisms that protect against scanning block or drop suspected packets, thus the attacker is aware that data is missing. Rather than alerting the attacker, Sniffer Reflector [49] obfuscates the responses to suspected packets, warping attackers' view of the network. Sniffer Reflector

utilises a scan sensor and shadow network to defend a protected network. The scan sensor scans all traffic and detects packets suspected of scanning. All the detected scanning packets are forwarded to the shadow network. The shadow network is a virtualised copy of the protected network although with port numbers and offered services modified. In this way, the shadow network can fool an attacker into believing that the received data is correct. The protected network is not aware of the scan sensor or shadow network and receives all packets that are not suspected by the scan sensor. The weaknesses of Sniffer Reflector are the scan sensor and randomisation of the network. The protection of the system relies on the scan sensor; thus, scans must be detected to be obfuscated. The shadow network remains similar to the real network but randomises port numbers and services, thus if a server does not display some essential services it can be easily discovered to be a fake.

2.3 Assessment of Network-Based MTD Techniques

The assessment of security techniques should be thorough including an assessment of both security and performance. Especially in the case of MTD techniques due to their relative youth in comparison to traditional security. Additionally, MTD adds extra complexities through dynamics. Ideally, the assessment of MTD involves realistic evaluation, bringing the benefit of discovering issues with existing infrastructure through implementation and deployment. The following describes the analysis of past MTD techniques along with the advantages and disadvantages of their approaches.

2.3.1 Topology Mutation Against Launch

Hong et al. [18] devised a shuffle-based MTD technique that randomly modified the topology of a network using SDN. The aim was to use a shuffle-based method to increase the diversity of a network by constructing diverse attack paths. The next network configuration is chosen from a set of possible configurations to maximise the number of variant Operating System (OS) nodes along all paths. Through simulations on a SDN testbed, the performance was evaluated, and the techniques security was analysed analytically. The overhead was measured in terms of packet loss rate and delay on delivered packets for different mutation intervals and topological changes. The lowest packet loss rate reported was less than 5% for a mutation interval of 1 second. Although in terms of the number of packets that are routed through a network this is massive. Reporting an acceptable packet loss rate, along with its corresponding mutation interval, would have increased confidence in the feasibility of the technique. Additionally, the security of the technique was not tested to ensure it matched the analytical model. In summary, the metrics used to study the performance overhead were chosen well but were not compared to typical packet loss rates.

2.3.2 Address Mutation

Address mutation techniques protect hosts by changing their identities proactively or reactively. Further, transparent approaches map between virtual and real address spaces. With these modifications, infrastructure incompatibilities can be introduced. For this reason, realistic evaluation is especially important. The following describes the analysis performed on past address mutation techniques.

Analysis of Limits to Address Shuffling

Many address shuffling techniques have been devised that employ address mutation [23, 29, 9] although there has not been a focus on the theoretical limits of address shuffling. Carroll et al. [4] does not propose a new MTD model but analyses the theoretical limits of address shuffling with a mathematical urn model. With the use of the urn model, this paper analyses how many probes an attacker needs before discovering a valid address. This condition would occur for an attacker or worm probing for a single open port for which they have an exploit. Two specific cases are modelled: static addresses and perfect address shuffling. In a traditional network, the addresses are static and therefore do not change over time. Perfect address shuffling simulates a reactive MTD technique where the address space is reshuffled after each of the attackers address probes. The test conditions in this paper are too broad such as assuming perfect shuffling and success of an attacker. Perfect address shuffling is a poor assumption since it is unattainable as the resultant overhead on the system would cause high latency and packet loss. Attacker success occurred if a single valid address was discovered although this did not consider the time for launching an attack.

Transparent Address Mutation Against Reconnaissance

The MTD techniques OF-RHM [23] and RHM [41, 22, 21] are incremental works on devising an address mutation technique that can defend against reconnaissance whilst meeting performance constraints. Over these works the analysis has grown although most are analytical or obtained through simulations, these techniques tend to miss real-world issues. The analysis contained includes performance overheads, network scans, and simulation

of scans. The measured performance overheads included analytical analysis of routing, mutation, DNS updates, and packet delay. The network scans were conducted with Nmap scans using host discovery, this did not involve testing various types of scans or testing the discovery of ports. The simulation of scans involved testing the techniques effectiveness against scanning worms, including the effect of worm propagation on scanning effectiveness. In summary, there is a lot of analysis here although most of it is analytical or simulation-based. These approaches are susceptible to missing real-world considerations including compatibility with common network infrastructure and software.

Non-Transparent Address Mutation Against Reconnaissance

The SDN shuffle [29] is a MTD technique designed to generate random synthetic IP and MAC addresses to defend against network reconnaissance. The synthetic IP and MAC are managed directly by the hosts rather than by a SDN controller and switches. The SDN shuffle was evaluated on performance and security. The performance was measured on virtualised machines using a software implementation of SDN Shuffle, specifically the latency introduced by the SDN controller handling Domain Name Service (DNS) requests and modification of NAT rules were measured. Processes such as address translation via the NAT device was not measured as the overhead has already been deemed acceptable due to its wide adoption. However, the address translation is usually applied by a NAT device rather than host systems, breaking this assumption. Additionally, some processes that lacked wide adoption such as generation or management of synthetic addresses was not analysed. Security was analysed analytically in terms of unpredictability, the vastness of the movement space, regularity of movement, and availability of protected services. Unfortunately, security was not realistically evaluated,

in a similar way to the performance, but rather a brief analytical review. In summary, the weaknesses of the analysis of SDN shuffle included ignoring possible performance delays and a lack of security testing on the implementation of SDN shuffle.

NASR [1] was an early address mutation technique that mutated addresses at a low rate using the DHCP. NASR tracks the TCP connections of protected hosts and coordinates address changes. With this and the abortion chance of TCP connections upon address changes, NASR had to be analysed. The analysis consisted of experimental and simulation approaches. Experimental analysis included host discovery over the Internet and disruption of TCP connections upon different address mutation intervals. The host discovery was conducted with application-level probing, random scanning over class B addresses, and search engine reconnaissance. The speed of randomisation was analysed to find how NASR would slow the spread of hitlist worms. The disruption of TCP connections was simulated with different address changing frequencies. The analysis considered many real-world considerations, for example, the experimental analysis was completed over the Internet and with massive Autonomous Systems (ASs). Unfortunately, the security analysis completed on NASR's mechanisms was simulated and delays related to switching addresses was not considered.

Address Mutation Against Reconnaissance and Access

In addition to address randomisation using vIP addresses, FRVM [42] multiplexes the addresses of hosts using service port numbers. This allows hosts to have multiple vIP addresses. The analysis consists of analytical security testing using a probability model to measure attacker success. An attack is assumed to be successful upon the discovery of a single host through its IP

address and modelled with a binomial model. Weaknesses in this analysis include lack of performance analysis, model assumptions, and missing multiplexing. The probability model is binomial which assumes each trial is independent, this is not true for scanning an address space unless address shuffling is performed after every probe. Otherwise the trials are not independent. Lastly, this analysis does not demonstrate the advantages of address multiplexing. In fact, discovering a single valid IP address will be easier with address multiplexing since there are more valid addresses. Analysis of port discovery should be shown to demonstrate the benefit of address multiplexing.

2.3.3 MT6D Based Techniques

MT6D [9, 15] mutates IPv6 addresses with a high frequency in comparison to other address mutation techniques. The testing involved both connectionless and connection-orientated traffic via Internet Control Message Protocol (ICMP) and TCP, respectively. For both traffic types, the packet loss and latency at each point in the system was recorded. This allowed the overhead of the system to be analysed, in addition to the specific components that caused bottlenecks. There appeared to be a lack of statistical evidence despite the comprehensiveness of the performance testing conducted such as missing confidence intervals for the collected metrics and no transparency in the number of samples collected for the system latency. Security testing consisted of analytical analysis of the address entropy. In summary, the scope of performance testing was impressive including measuring delays at multiple points in the system although necessary statistical analysis was not conducted. Additionally, security testing was lacking a thorough analysis of the level of security provided by MT6D.

The MTM6D [17, 16], designed with the concepts of MT6D [9] in mind, applies address mutation to VPN servers as an anti-censorship technique. The technique's performance was analysed analytically and experimentally with respect to packet overhead and packet loss, respectively. The packet overhead strictly recorded the additional packet headers and their sizes in bytes, that were needed for the technique. Packet loss between movement of addresses was measured in a testbed, this was compared without MTD and at different addresses movement intervals. Unfortunately, some analysis was missing including the performance impact to ongoing connections and analysing the security benefit. The extra performance cost to ongoing connections in terms of latency or speed was not recorded. The security benefit with regards to tracking attacks was not analysed. Despite deployment of MTM6D on a physical network, there has not been a large amount of formal analysis, rather the deployment was in the aim of testing optimisations in a realistic setting.

2.3.4 Obfuscation of Network Information

Chaos [43] is a MTD technique designed to detect threat levels and obfuscate unexpected traffic accordingly in terms of IP address, port numbers, and system fingerprints. Chaos was evaluated in terms of performance and security and compared against an unprotected system and MTD protected system with static obfuscation. Security of the three networks was evaluated using Nmap [28]. The techniques were compared with respect to the level of information disclosure. This revealed how effective Chaos was at detecting the threat level and applying the appropriate level of obfuscation. Unfortunately, the variability of the collected data was not demonstrated. Performance of

the technique was verified by measuring the average packet delay in the network, although the exact paths were unspecified. In summary, comparisons were made between networks but statistical tests to prove the integrity of the data were not performed.

Sniffer Reflector [49] attempts to obfuscate reconnaissance and access by detecting and responding with incorrect information to network scans. The proposed system includes a scan sensor, shadow network, and a protected network. A software implementation of Sniffer Reflector was deployed in a visualised environment to access its security and performance. Security was accessed by performing network scans via Nmap [28] and comparing the collected information against a network without Sniffer Reflector deployed. Success was measured by the level of obfuscation of the scanned network details. Performance testing was lacking except for latency introduced at the attacker end whilst scanning. Although the performance effect on users was not analysed. Security testing was performed and included comparisons to typical systems but without statistical integrity. There appeared to be a sample size of one for the Nmap scans and statistical rigour between the different scanned systems was missing leading to a lack of confidence in the collected results. In summary, the performance and security testing conducted had some significant issues. Although testing with a typical system to compare results was well founded, as it provided a baseline for the results.

2.4 Summary

Design of MTD techniques has matured and general guidelines for this process are available. The current material introduces models along with evaluations of past MTD techniques to allow future designers to improve on the

past. Unfortunately, the focus of MTD evaluation tended towards model-based approaches with little attention towards realistic evaluation. Further, if analysis contained realistic evaluation, usually it ignored either performance or security. Realistic evaluation has the benefit of testing a MTD technique alongside existing infrastructure. Specifically of interest to this thesis are network-based MTD techniques that protect a host by obscuring their identity. FRVM, the focus of this thesis, belongs to this group. Most address mutation techniques can operate transparently to the protected hosts although the method to obtain transparency differs. The techniques used include a dedicated gateway that intercepts traffic and encapsulates the packets and a SDN setup that maps between address spaces. Some MTD attempt to make performance savings by applying dynamics only to those hosts that are suspected of being an attacker. Unfortunately, detection will always have false negatives, thus unknown or stealthy scanning techniques can operate undetected on these networks. These issues and the need for security and performance comparisons between competing MTD techniques give rise to the need for thorough security and performance testing. Many MTD techniques, despite describing a software implementation, did not perform realistic evaluation with respect to either security or performance. The evaluation of results for MTD techniques reviewed usually lacked important steps such as including a control group or evaluating data with statistical rigour.

Chapter 3

Method

Flexible Random Virtual internet protocol Multiplexing (FRVM) is a recent Moving Target Defence (MTD) technique that defends against attacker network reconnaissance and access. The defence involves mutating multiple Internet Protocol (IP) addresses per protected hosts on an interval. FRVM has been analytically analysed in terms of security [42]. Although, FRVM is lacking realistic evaluation to verify the analytical analysis and identify ambiguities and issues through the design, implementation, and deployment process. These problems lead to the following research questions.

- What extra considerations will need to be handled to implement and deploy FRVM?
- How does FRVM affect security scanning in terms of information disclosure and scan duration?
- What is the effect of FRVM on a network's performance?

In the conquest of answering these questions, a methodology was formed. This chapter outlines the components of realistically evaluating FRVM: threat model, FRVM theory along with its benefits and issues, metrics of analysis, configuring the virtualised network, implementation of SDN controllers,

measurement instruments, data collection, and data analysis. The threat model describes the assumptions and methods that the modelled attacker would use in their attempt to bypass security. The security and performance of FRVM was evaluated with network scanning and file transfer duration over the Transmission Control Protocol (TCP), respectively. Further, typical Software Defined Network (tSDN) and Random Simplex vIP Mapper (RSM) controllers were evaluated alongside FRVM for comparison. Measurements of the SDN controllers were taken on a virtualised network. Security was measured with Nmap and judged by the level of information disclosure and scan duration. Performance was measured with TCP file transfers at varying levels of network load, judged by the duration of the transfer. The quantitative data collected from security and performance testing was analysed with inferential and descriptive statistics.

3.1 Threat Model

Before the security of FRVM could be evaluated, knowledge of the methods an attacker could employ to bypass FRVM had to be known. The implicit assumption is that an attacker needs to complete these before exploiting a system, the assumed attack chain follows [30].

$$\text{Reconnaissance} \rightarrow \text{Access} \rightarrow \text{Development} \rightarrow \text{Launch} \rightarrow \text{Persistence} \quad (3.1)$$

Firstly, the attacker must perform reconnaissance to locate the host via an IP address. Access follows from reconnaissance involving the discovery of detailed information about the target, like the open ports on which vulnerable services operate. Development utilises the detailed information found in the access stage to craft an attack. The attack is then realised in the launch phase.

Persistence involves adjusting the exploited system to retain hold. Clearly, the completion of any link assumes the successful completion of all previous links. FRVM protects against reconnaissance and access through address mutation and multiplexing, stopping attackers before they gain a foothold.

3.2 Overview of FRVM

FRVM [42] is a shuffle-based MTD technique that mutates and obscures the attack surface of a network. It achieves this by periodically changing the IP addresses of protected hosts and allocating IP addresses per service. With SDN, FRVM can intercept network traffic and modify network switches flow tables to dynamically change the behaviour of a network. FRVM's address mutation and multiplexing protects hosts from attackers. Hosts and their services are identified with IP addresses and port numbers, respectively. Rather than mutating the IP addresses of hosts directly, FRVM introduces virtual IP addresses that allow IP address mutations to act transparently from the protected hosts. Every protected host retains their real IP (rIP) address known by themselves but in addition, hosts have virtual IP (vIP) addresses known and managed by the FRVM controller. Protected hosts are identified using their vIP addresses rather than their rIP address. Switches connected to the controller change between rIP addresses and vIP addresses whilst routing, allowing address changes to be transparent to the protected hosts. The transparency avoids extra overhead on the protected hosts as the FRVM controller and switches manage the vIP addresses and the mapping between them. Periodic address mutation in the virtual address space provides additional protection to hosts as their identity changes making it difficult for attackers to track the hosts.

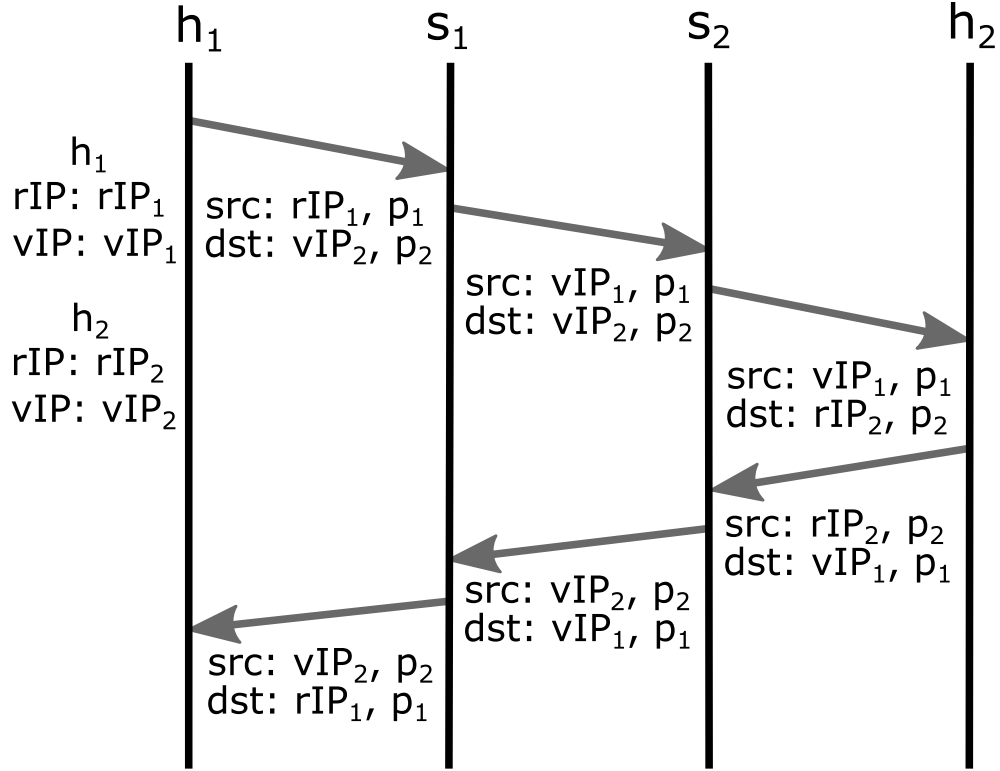


FIGURE 3.1: Illustrates communication between two FRVM protected hosts in the same subnet, h_1 and h_2 , that are mapped and forwarded by two intermediate switches, s_1 and s_2 .

During reconnaissance and access, an attacker attempts to learn the IP address of a host and find the open ports that operate exploitable services. The mutation of IP addresses defends against attacker reconnaissance by increasing the difficulty of locating hosts and storage of discovered host information. The multiplexing of IP addresses minimises the damage of IP discovery by addressing every service with its own IP address. Further, the danger of system fingerprinting is abolished as each IP address exposes only a partial view of the host. Since network reconnaissance and access are often the first two steps of an attack [46], FRVM can protect against unknown vulnerabilities.

3.2.1 Address Mapping

FRVM protects hosts from network reconnaissance by periodically changing vIP addresses, thus stopping attackers from gaining a foothold in the system. Similar to earlier techniques [23, 41, 22, 21], FRVM maps rIP addresses to vIP addresses at the network edge by mapping IP addresses at the closest switch to a protected host, demonstrated in Figure 3.1. An edge switch is the closest intermediate switch on the hosts network path, in the Figure 3.1 s_1 and s_2 are the edges switches for h_1 and h_2 , respectively. Earlier techniques have applied vIP addresses to avoid attacker reconnaissance although FRVM additionally protects against access through addressing each service offered by a protected host with a unique vIP address and port number pair, referred to as vIP address multiplexing. FRVM protected hosts have a single rIP address but can have many vIP addresses. Additionally, a vIP address may be associated with multiple hosts through different port numbers. Address multiplexing further obfuscates the identity of hosts changing the view of a network and the services offered. Formally, FRVM has m -to-1 multiplexing and 1-to- m de-multiplexing between a real address space and virtual address space, respectively, where m is the number of services on each host. The domain and co-domain of the multiplexing and demultiplexing functions can be expressed as

$$f_{\text{multiplexing}} : \mathbf{VIP} \times \mathbf{P} \rightarrow \mathbf{RIP} \times \mathbf{P} \quad (3.2)$$

and demultiplexing

$$f_{\text{demultiplexing}} : \mathbf{RIP} \times \mathbf{P} \rightarrow \mathbf{VIP} \times \mathbf{P} \quad (3.3)$$

using the sets

$$\mathbf{RIP} = \{\text{rIP}_1, \text{rIP}_2, \dots, \text{rIP}_R\} \quad (3.4)$$

and

$$\mathbf{VIP} = \{\text{vIP}_1, \text{vIP}_2, \dots, \text{vIP}_V\} \quad (3.5)$$

and

$$\mathbf{P} = \{p_1, p_2, \dots, p_P\} \quad (3.6)$$

where \mathbf{RIP} , \mathbf{VIP} , and \mathbf{P} are the sets of rIP addresses, vIP addresses, and port numbers, respectively. It follows that the multiplexing and demultiplexing functions can be expressed as

$$f_{\text{multiplexing}}(\text{vIP}_i, p_i) = (\text{rIP}_j, p_i) \quad (3.7)$$

$$f_{\text{demultiplexing}}(\text{rIP}_j, p_i) = (\text{vIP}_i, p_i) \quad (3.8)$$

where $f_{\text{multiplexing}}$ is the inverse of $f_{\text{demultiplexing}}$. Note that the port numbers associated with a rIP and vIP mapping remain static.

3.2.2 Benefits of FRVM

FRVM is not the first MTD technique to introduce address mutation, or even achieve transparency using vIP addresses. FRVM introduces address multiplexing that provides benefits on top of past techniques including increased defence against host discovery methods, less static information between movements, and increased search space.

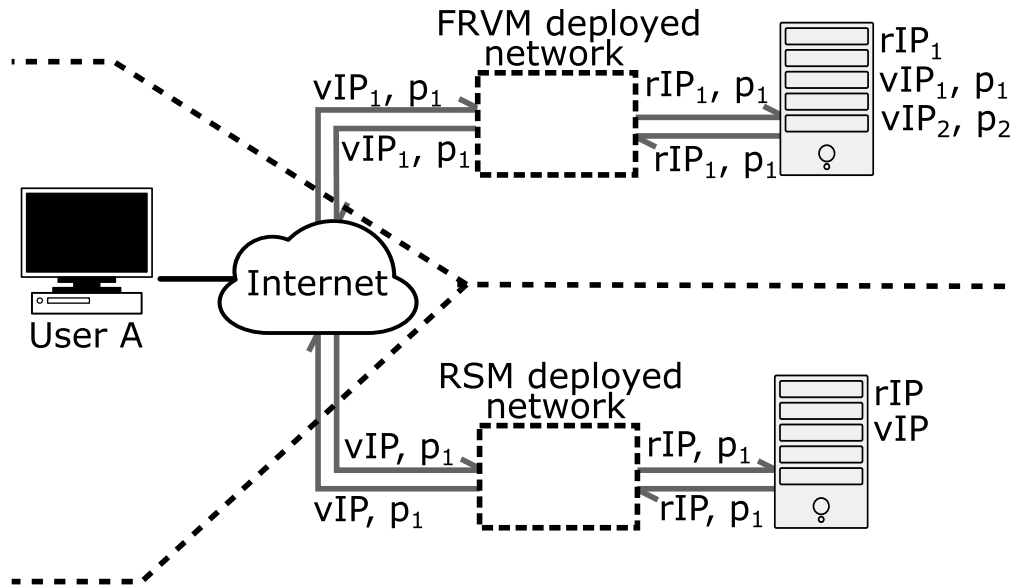


FIGURE 3.2: Illustrates the difference made by address multiplexing with respect to information disclosed by a port probe. The comparison is between RSM and FRVM deployed networks. User A probes a host in RSM and FRVM networks.

Defence Against Host Discovery

Host discovery is the first step of a network scan, in which probes are sent to potential hosts to check their availability. Typically, this reduces the amount of work in the next step, port discovery, by decreasing the number of hosts. Common probes include Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), User Datagram Protocol (UDP), and TCP; the transport layer probes are sent to commonly open ports. In an unprotected network, a host will answer to any of these probes. Although, FRVM reduces the information disclosed by changing the network view of hosts through address multiplexing. With address multiplexing, if a host responds to a probe only the virtual address associated with the probe's port is exposed to the attacker. The differences are summarised in Figure 3.2, this shows that after probing FRVM, the host only knows the vIP address that communicates with the probed port. However, probing a controller that employs address mutation, without address multiplexing, still results in an address that could be

used for communication with any port on the host.

Although not immediately clear, address multiplexing protects FRVM from outside attackers attempting host discovery, increasing the search space of the scan. Between networks, attackers can probe with ICMP or common ports through UDP and TCP. Firstly, IP multiplexing requires the use of a port number in addition to the IP address. Although ICMP is a portless protocol as it operates below the transport layer. Therefore, ICMP packets will not be mapped to a virtual IP address. Secondly, using common port numbers doesn't allow attackers to discover hosts as FRVM addresses each service rather than each host.

Reduced Static Information of FRVM

Attackers cannot identify hosts between different scans due to FRVM's protection. Without address multiplexing, after scanning a host the set of open ports discovered through a single IP address can be used to fingerprint a host, even with address mutation protections. Address multiplexing stops fingerprinting by addressing each service of a host, rather than each host. Therefore after scanning, the set of discovered open ports are each associated with different IP addresses, with no way to correlate them to a single host.

Increased Search Space of FRVM

The search space for a network scan on FRVM is largely increased due to its defence against host discovery through multiplexing over ports. Without host discovery, all port probes must be sent to every address in the subnet range. Usually there are many more port probes than host probes, therefore this dramatically increases the duration of a scan.

3.2.3 Issues with Existing Network Infrastructure

The specification of FRVM contains ambiguities that cause problems during integration alongside existing network infrastructure. These include TCP incompatibility, router address randomisation, and multiplexing over portless protocols. These problems are discussed below and their resolutions for the FRVM software implementation in Section 3.6.

TCP Connections

Currently, FRVM is incompatible with TCP due to address mutations. TCP connections make the implicit assumption that the communicating hosts IP addresses will remain static during the connection, an assumption that the mutation of IP addresses violates. It follows that if an FRVM address mutation occurs during an ongoing connection, the communicating hosts are disconnected.

Default Gateways

Hosts typically rely on a default gateway to route packets towards destinations outside of the hosts' network. This greatly simplifies the task of routing for the host, allowing the host to define a single static IP address route for all outside hosts. With FRVM's address mutation, a routers IP address would not be static. It follows that if a router is allocated a vIP address, the hosts cannot make use of default gateways.

Portless Protocols

FRVM multiplexes vIP address and port number pairs into corresponding rIP address and port number pairs, as described in Subsection 3.2.1. However, this makes the implicit assumption that every packet has a port number associated with it although this is not the case. Example protocols that operate without the transport layer include ARP and ICMP. ARP is a request-response protocol used to discover a host's corresponding Media Access Control (MAC) address given the IP address is known, a function that is critical for communication. The request is often broadcast over the network and the host with the matching IP address responds with their MAC address. Although without a port number, FRVM cannot multiplex over the IP addresses.

3.3 Metrics of Evaluation

The evaluation of FRVM was split into two parts: security and performance. Demonstrating the effect of a SDN controller on either requires metrics that reliably and accurately describe the effect. The metrics selected for security and performance testing are described separately in the following subsections.

3.3.1 Metrics of Security Testing

The security of the tSDN, RSM, and FRVM controllers was demonstrated through network scans with Nmap [28] involving host and port discovery. Similar techniques [49, 22, 1] have utilised network scanning to model security. Network scans provided metrics for the number of hosts discovered,

number of ports discovered on each machine, and duration of the scan. The former two were collected as information disclosure metrics.

The primary aim of attackers scanning is to obtain information about the target network that can be used to develop and launch an exploit. A variety of information is collected that describes the hosts including availability, ports along with statuses, and the Operating System (OS). The host discovery phase determines the availability. The port discovery phase determines the open, filtered, and closed ports and the OS of the host if fingerprinting was successful. Neither the filtered or closed ports, nor the OS fingerprint, were used to judge the level of information disclosure. Ports that are marked as filtered or closed are not exploitable, unlike open ports. The OS of the host was not included as detecting this incurs extra duration. Due to the address mutations of the MTD controllers, the duration of scans could significantly affect the success of network scans. Scans both with and without fingerprinting were not taken due to time constraints of the thesis. It follows that the amount of information disclosure was measured with respect to discovered hosts and ports, to represent the success of an attacker in reconnaissance and access, the links protected by FRVM.

FRVM mutates IP addresses on an interval, hiding addresses from attacker and expiring any collected information. It follows that the duration of a scan can affect the amount of collected data that is still valid after completion, due to address mutations. Further, address mutation and multiplexing add additional difficulties to scanning. These difficulties include large packet timeouts waiting on a host's response that is no longer associated with the probed IP address. These difficulties can be measured by duration of the scan.

3.3.2 Metrics of Performance Testing

The duration of file transfers over TCP, for differing levels of network load, demonstrated the performance of the tSDN, RSM, and FRVM controllers. The load was varied separately by either increasing the number of concurrently transferring servers or connections to a server.

The goal was to find a metric that described the additional delay experienced by the network due to the deployment of the SDN controllers. This could either be measured by finding the extra delay experienced by packets through the network, or the delay incurred by extra actions performed by the controllers. The former was chosen over the latter because of the difficulty in measuring small delays on the virtualised network as differences could be obscured by measurement error. Specifically, the duration of file transfer over TCP measured the accumulated additional delay over the packets in the connection. Additionally, this allowed the load to be varied with more connections. Delays have been measured with connection orientated traffic by a past address mutation technique [10], although without the addition of varying levels of load.

3.4 System Software

All the development software used in the evaluation of FRVM is freely available, with a large portion open source, improving the replicability of the experiment. The development environment operated Ubuntu 17.10 64-bit [27] virtualised with Oracle VM VirtualBox [33], as a subset of the development software required a Linux distribution. All program logic was implemented with Python [36], due to its readability and far reach. Python 2.7, rather than

Python 3.x, was required by a subset of the development libraries. Python has far reach because it is taught to many modern developers as their first programming language and ranks among the top programming languages in popularity and usage [13, 8]. The software implementations of tSDN, RSM, and FRVM operated on Software Defined Networking (SDN) controllers that were deployed onto a virtualised network. Essential software for the virtualisation of FRVM included Ryu [37] and Mininet [26]. Ryu is a Network Operating System (NOS), written in Python, that provides Python libraries for OpenFlow communication. Mininet can create virtualised networks containing nodes such as hosts, OpenFlow switches, and SDN controllers using network namespaces and virtual Ethernet pairs [26]. Additionally, Wireshark [12] was used to sniff packets on the virtualised network to aid in controller debugging and discovery of large delays. The software implementations of the tSDN, RSM, and FRVM were measured in terms of security and performance. The resultant metrics were analysed using inferential and descriptive statistics, using a statistical computing platform, R [35]. The development environment and additional libraries are summarised, along with versions, in Table 3.1.

3.5 Network Configuration

For the realistic evaluation of FRVM, the controller was deployed onto a virtualised network, constructed with Mininet [26]. The security and performance of the tSDN, RSM, and FRVM controllers was measured on the virtualised network. Mininet provides the ability to create large virtualised networks that are portable, do not require expensive hardware, and are easily shareable for others to confirm results. Off the shelf, Mininet provides pre-made network elements such as hosts, OpenFlow switches, and SDN

TABLE 3.1: Software specifics of the development environment for the thesis.

Detail	Info
Hypervisor	Oracle VM VirtualBox 5.2.12
Operating System	Ubuntu 17.10
Central Processing Unit	Intel i7-3770K @ 4.20 GHz
Programming language	Python 2.7.14
Data processing language	R v3.5.1
Network virtualisation	Mininet v2.2.2
SDN Controller software	Ryu v4.25
OpenFlow version	1.2
OpenFlow switch version	2.8.1
Network scanner	Nmap 7.60
Packet dissector	Wireshark 2.4.2
Integrated Development Environment	PyCharm Community 2018

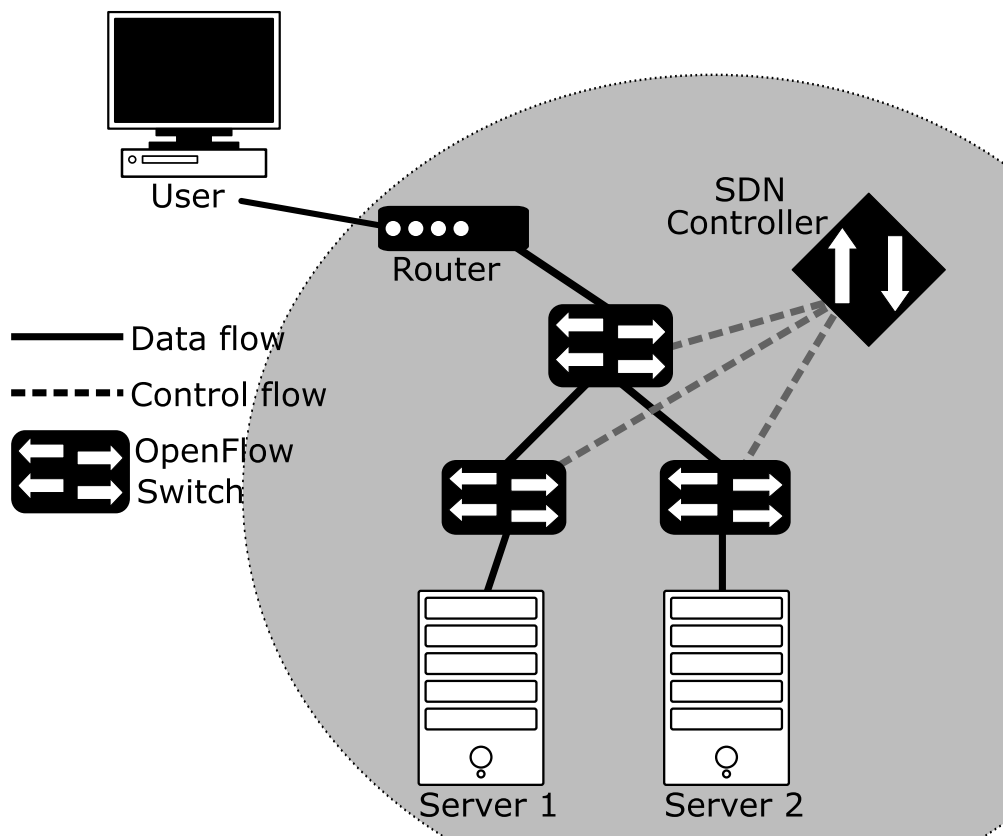


FIGURE 3.3: Illustrates the basic topology of the virtualised network containing a user, router, switches, SDN controller, and protected servers. A key is given on the left to separate the links used for data flow and control flow.

controllers. All network elements run a Linux sub-system that operates real-world protocols and applications. Alternatives to Mininet included an abstracted network platform that simulated relevant network mechanisms, or a physical SDN-based network. The abstracted network platform would not have captured how FRVM effects existing real-world network infrastructure, therefore, avoiding a realistic implementation of FRVM. Whereas, FRVM was not deployed on a physical SDN-based network due to time constraints related to the configuration of physical network elements. Mininet was preferred over both options due to the ease of configuration and the use of real-world applications, protocols, and off the shelf network elements.

The basic topology of the virtualised network is shown in Figure 3.3. The network contained a router, outside user, SDN controller, OpenFlow switches, and protected servers. The router divided the two networks, one containing an outside user and the other a SDN-based network. The division of networks simulated the outside user connecting from the Internet. The tSDN, RSM, and FRVM controllers were each deployed in a network with this topology. The protected servers were unaware of the SDN controller's operation. Small variations on this network were used for measuring security and performance of the tSDN, RSM, and FRVM controllers. The alterations made to the network for security and performance testing are described in the following subsections.

3.5.1 Modifications for Security Testing

Security is a large concern for FRVM, the virtualised network needed to be capable of testing reconnaissance and access of an outside attacker. For security testing, the outside user become an outside attacker configured to use a network scanner, Nmap [28]. The links were configured with artificial delays

to simulate roundtrip delay. The internal and external links were configured with 1ms and 30ms delays, respectively.

3.5.2 Modifications for Performance Testing

The performance of the tSDN, RSM, and FRVM controllers was measured with the duration of TCP file transfers over different levels of network load. Measuring the durations required minor modifications to the virtualised network such as the addition of more hosts and bandwidth restrictions on the simulated links. Additional hosts were added both outside and inside of the network. The hosts were connected to the network via the switches, in a round-robin with the switches, although the number of switches was held constant at three. The bandwidth on all the internal network links was restricted to 100Mb/s whereas the outbound link was restricted to 10Mb/s. The outbound links were restricted to a lower speed than internal links to simulate the speed difference between Internet connections and Local Area Network (LAN) bandwidth. The links were restricted to ensure the TCP acted normally within the virtualised network. Normally the TCP needs to deal with issues such as random delays and packet loss although these were not modelled by the virtualised network, due to their inherent randomness obscuring data.

3.6 SDN Controller Implementations

The aim of the thesis was to evaluate a software implementation of FRVM in realistic conditions and compare it with pre-existing analytical results. Therefore, FRVM was implemented with Ryu [37], a NOS, and deployed on a virtualised SDN controller. Additionally, the results collected from FRVM were compared with results from the tSDN and RSM controllers, requiring a software implementation of both.

Alternative solutions to Ryu for communicating through OpenFlow include implementing OpenFlow communication or utilising one of the many available NOSs. Firstly, the use of a pre-implemented OpenFlow communication was a necessity due to the complexity of the OpenFlow protocol and time restrictions of this thesis. Secondly, the chosen NOS had to provide a Python API for ease of development and support higher versions of OpenFlow to allow expansion of FRVM. There are alternatives that meet these restrictions other than Ryu such as Floodlight [11]. Ultimately, Ryu was chosen for its complete documentation, open source codebase, and additional helper libraries including a packet decoder for major network protocols.

Ryu handled setup configurations such as the OpenFlow handshake with the OpenFlow switches and provided an API with methods for communication between the controller and switches with the OpenFlow protocol. Important OpenFlow communication packets included packet-in, packet-out, and flow-mod [32]. Packet-in is sent from a switch to the controller in the event of a flow table miss in the sending switch. A flow table miss occurs if a packet arrives at a switch but there is no forwarding table rule that matches the arrived packet. Packet-out contains a packet for the data plane in its payload and is sent from a controller to a switch instructing the switch to forward the packet out through a specified switch port. Packet-out allows the controller

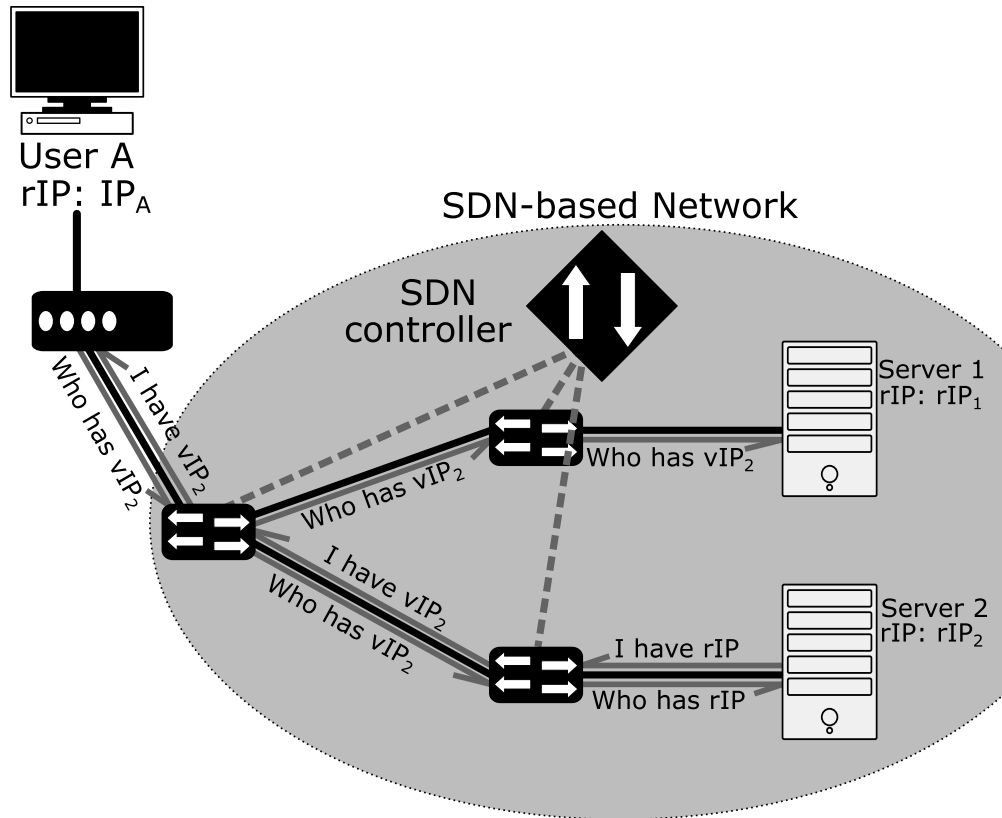


FIGURE 3.4: Demonstrates how an ARP exchange spreads through a FRVM deployed network, specifically how the vIP address is mapped at network edge of server 2.

to route packets through switches. Flow-mod is an essential configuration packet that allows the controller to install new flow rules in switches, thus installing missing flows and facilitating routing in a network. Specifics of the implementations of the three controllers are outlined in the following subsections.

3.6.1 FRVM Controller

The FRVM controller is a software implementation of the MTD technique described by Dilli et al. [42] and Section 3.2. FRVM implements transparent

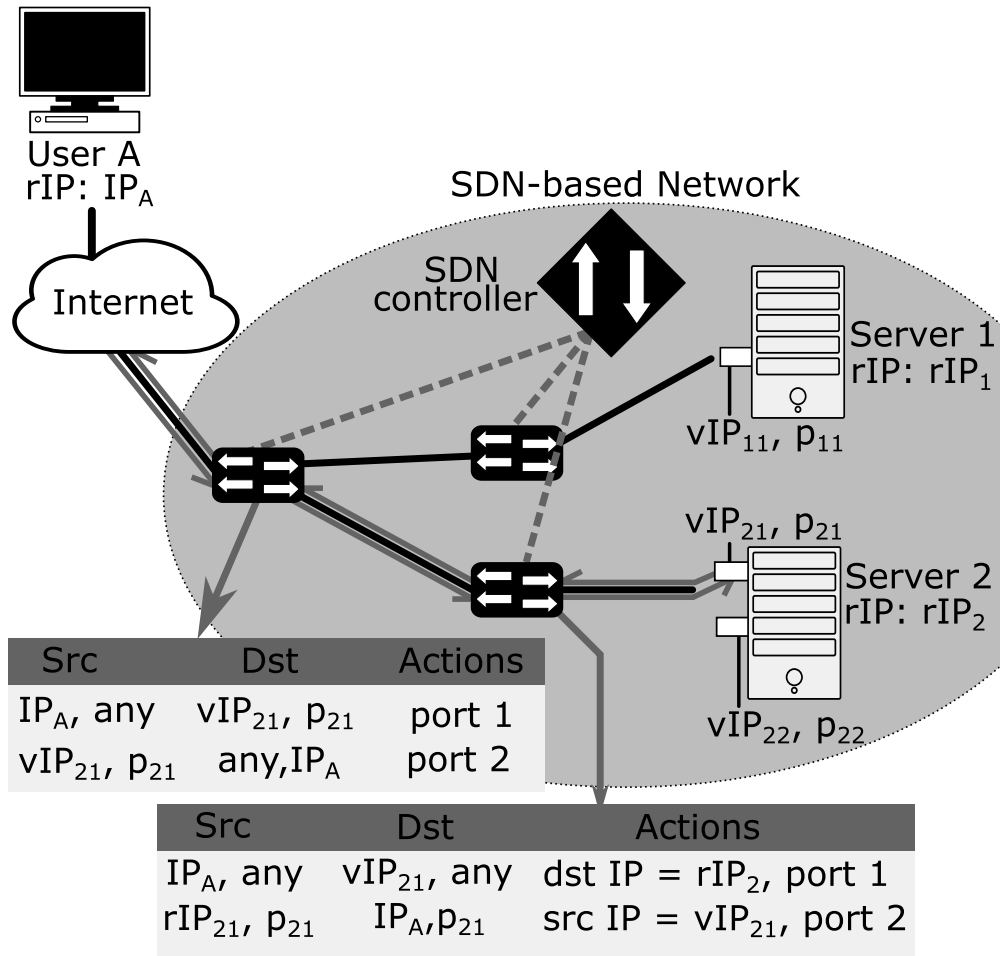


FIGURE 3.5: Demonstrates the contents of switch flow tables after routing a packet between user A and Server 2 for a FRVM deployed network.

Algorithm 1: Pseudo code for the FRVM controller's vIP address generation.

```

Allocations = {};
for all hosts  $h$  in protected network do
  for all services  $s$  for  $h$  do
    do
      Generate  $vIP$ ;
      while ( $vIP, s.port$ ) not in  $Allocations.vIP_{pairs}()$ ;
      Allocations[( $h.rIP, s.port$ )] = ( $vIP, s.port$ );

```

Algorithm 2: Pseudo code for the FRVM controller.

```

Generate vIP addresses for all hosts;
Create timer for next address mutation;
for all Packet-in packets  $p$  from OFswitches do
    if  $p$  contains an ARP or IP packet from  $h_i$  to  $h_j$  then
        if OFswitch is edge switch for  $[h_i, h_j]$  then
            Install in flow in src OFswitch
            action set  $srcIP(p) := vIP(h_i, p.src\_port)$  and
                 $dstIP(p) := rIP(h_j, p.dst\_port)$ ;
            Install out flow in src OFswitch
            action set  $srcIP(p) := p.dst\_IP$  and
                 $dstIP(p) := pkt.src\_IP$ ;
            Output  $p$  in Packet-out to src OF-switch;
        else if OFswitch is edge switch for  $h_i$  then
             $p.src\_IP = vIP(h_i)$ ;
            Output  $p$  in Packet-out to src OF-switch for flooding;
        else if OFswitch is edge switch for  $h_j$  then
            Install in flow in src OFswitch
            action set  $srcIP(p) := p.dst\_IP$ 
            Output  $p$  through OFswitch input port;
            Install out flow in src OFswitch
            action set  $dstIP(p) := rIP(h_j, p.dst\_port)$ ;
            Output  $p$  in Packet-out to src OF-switch through  $h_i$ 
                connected port;
        else if  $p.dst\_IP$  is not  $rIP$  and  $p.dst\_IP$  is  $vIP$  then
            Install in and out flow in src OFswitch;
            Output  $p$  in Packet-out to src OFswitch flooded;
    if timer has elapsed then
        Create timer for next address mutation;
        Generate vIP for all hosts and port combinations;
        Expire current flows;

```

IP address mutation and multiplexing. Address mutation involves periodically changing vIP addresses of hosts. Address multiplexing allows multiple vIP addresses to be assigned per rIP address, by using port numbers to discern between the multiple vIP addresses. It follows that FRVM must match packets on the network and transport layers to enable address mutation and multiplexing. The vIP address generation is detailed in Algorithm 1, where random numbers are generated with a cryptography secure method using randomness collected by the Operating System (OS). FRVM Modifies ARP and IP packets. Specifically, source and destination IP address fields of ARP and IP packets are modified by the FRVM controller and switches at the network edge. A high-level interpretation of the implemented FRVM technique is shown Algorithm 2.

Solutions for Existing Network Infrastructure

Whilst creating a software implementation of FRVM, ambiguities in the technique specification [42], as described in Subsection 3.2.3, needed solutions before a software implementation could be deployed on a network with existing network infrastructure. These ambiguities included router address randomisation and ARP address randomisation.

Routers in the FRVM protected network were not allocated vIP addresses. This meant the routers were contacted through their rIP addresses and the protection of FRVM was not afforded to the routers. Although as routers stitch networks together, they would be reachable through an interface on the connected non-protected network, therefore, the loss of protection is minimal. Without a vIP address, the routers have a static IP address allowing hosts within the network to use them as default gateways.

Algorithm 3: FRVM's multiplexing and demultiplexing with ARP, a portless protocol.

```

Initialise ARP_stack;
for all Packet-in packets  $p$  from OFswitches do
    if  $p$  is a ARP packet from  $h_i$  to  $h_j$  then
        if OFswitch is edge switch for  $h_i$  and ARP is a response then
             $p.src\_IP = ARP\_stack.pop()$ ;
        else if OFswitch is edge switch for  $h_j$  and ARP is a request then
             $ARP\_stack.push(p.dst\_IP)$ ;

```

ARP translates MAC addresses to IP addresses within a network, this functionality is necessary for end-to-end communication. Ideally, ARP packets would be treated in a similar fashion to IP packets, multiplexing and demultiplexing between vIP and rIP addresses, as shown in Figure 3.4. Unfortunately, multiplexing relies on the inclusion of port numbers, of which ARP packets have none. The solution involved implementing a queue for each host to identify which vIP address of a host, an ARP response should be translated to at networks edge. The technique is detailed in Algorithm 3. This solution could also be applied to other portless protocols like the ICMP.

3.6.2 tSDN Controller

Algorithm 4: Pseudo code for the tSDN controller.

```

initialise mac_to_port;
for all Packet-in packets  $p$  from OFswitches do
    if  $p$  contains an Ethernet frame from  $h_i$  to  $h_j$  then
        Add  $p.eth\_src$  to mac_to_port for OFSwitch;
        if  $p.eth\_dst$  in mac_to_port for OFSwitch then
            Install it and out flow in OFswitch;
        else if OFswitch is edge switch for  $h_i$  then
            Output  $p$  in Packet-out to OF-switch for flooding;

```

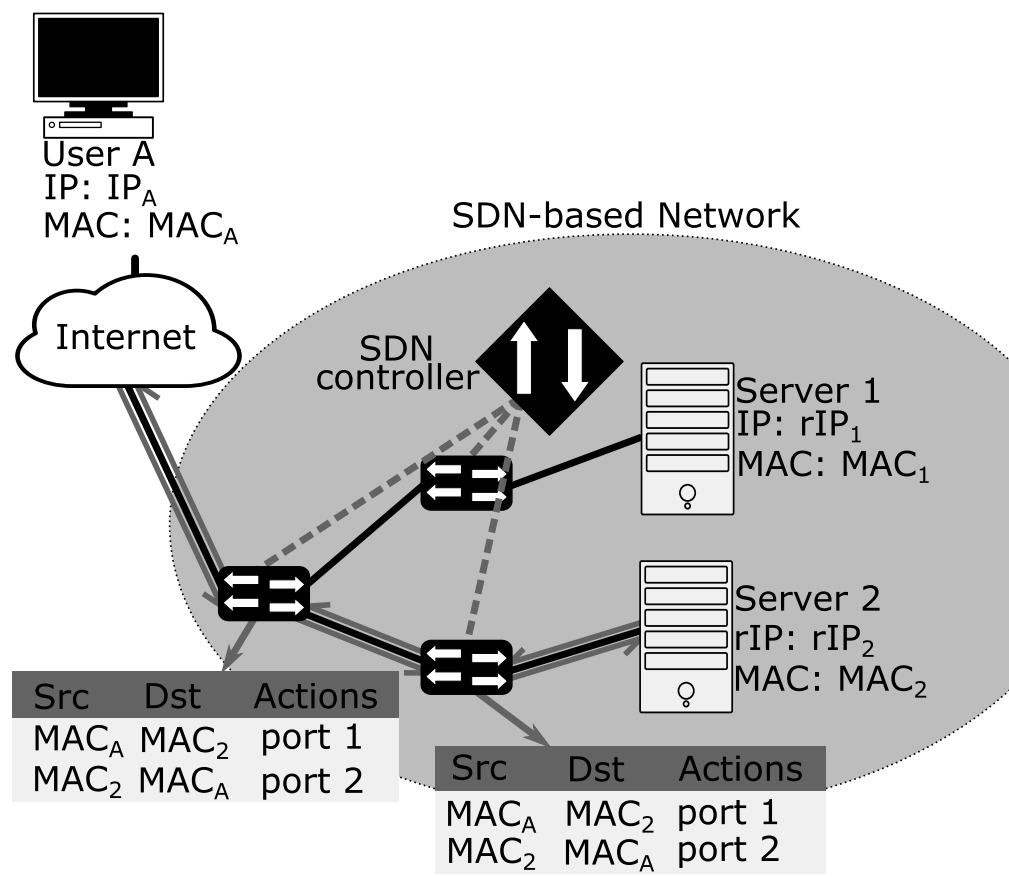


FIGURE 3.6: Demonstrates the contents of switch flow tables after routing a packet between user A and Server 2 for a tSDN deployed network.

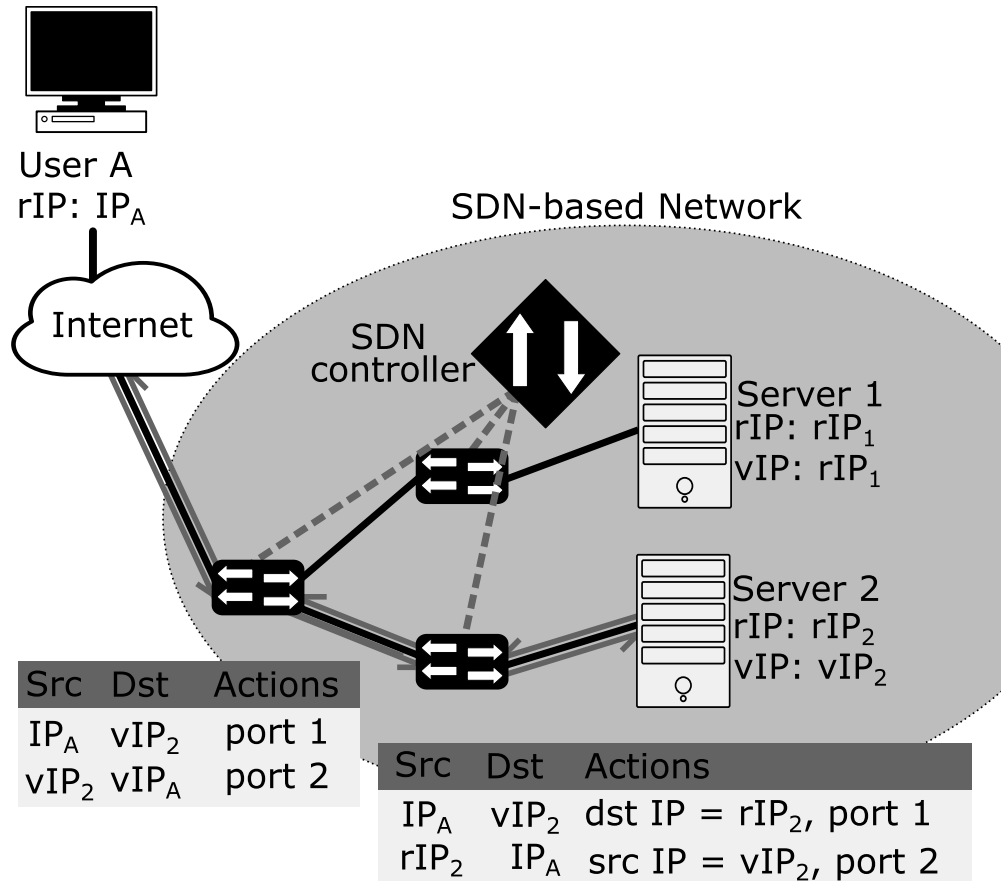


FIGURE 3.7: Demonstrates the contents of switch flow tables after routing a packet between user A and Server 2 for a RSM deployed network.

The tSDN controller implemented Ethernet-level routing on a SDN-based network. The tSDN controller only modifies Ethernet packets and the configuration is static, avoiding much of the complexity of both the MTD controllers. A high-level interpretation of the tSDN controller is shown in the Algorithm 4.

3.6.3 RSM Controller

The RSM controller implemented an address mutation technique similar to FRVM, although protected hosts have a single vIP address rather than many. Since RSM has a single vIP address per host, the simplex mapping functions

Algorithm 5: Pseudo code for the RSM controller.

```

Generate vIP for all hosts;
Create timer for next address mutation;
for all Packet-in packets p from OFswitches do
    if p contains an ARP or IP packet from  $h_i$  to  $h_j$  then
        if OFswitch is edge switch for  $[h_i, h_j]$  then
            Install in flow in src OFswitch
            action set  $srcIP(p) := vIP(h_i)$  and
             $dstIP(p) := rIP(h_j)$ ;
            Install out flow in src OFswitch
            action set  $srcIP(p) := p.dst\_IP$  and
             $dstIP(p) := pkt.src\_IP$ ;
            Output p in Packet-out to src OF-switch;
        else if OFswitch is edge switch for  $h_i$  then
             $p.src\_IP = vIP(h_i)$ ;
            Output p in Packet-out to src OF-switch for flooding;
        else if OFswitch is edge switch for  $h_j$  then
            Install in flow in src OFswitch
            action set  $srcIP(p) := p.dst\_IP$ 
            Output p through OFswitch input port;
            Install out flow in src OFswitch
            action set  $dstIP(p) := rIP(h_j)$ ;
            Output p in Packet-out to src OF-switch through  $h_i$ 
            connected port;
        else if  $p.dst\_IP$  is not  $rIP$  and  $p.dst\_IP$  is  $vIP$  then
            Install in and out flow in src OFswitch;
            Output p in Packet-out to src OFswitch flooded;
    if timer has elapsed then
        Create timer for next address mutation;
        Generate vIP for all hosts;
        Expire current flows;

```

Algorithm 6: Pseudo code for RSM controller's vIP generation.

```

Allocations = {};
for all hosts h in protected network do
    Generate  $vIP$  not in Allocations.vIPs();
    Allocations[rIP(h)] =  $vIP$ ;

```

domain and co-domain are

$$f_{simplex} : \mathbf{VIP} \rightarrow \mathbf{RIP} \quad (3.9)$$

and de-simplex

$$f_{de-simplex} : \mathbf{RIP} \rightarrow \mathbf{VIP} \quad (3.10)$$

using the sets

$$\mathbf{RIP} = \{rIP_1, rIP_2, \dots, rIP_R\} \quad (3.11)$$

and

$$\mathbf{VIP} = \{vIP_1, vIP_2, \dots, vIP_V\} \quad (3.12)$$

where \mathbf{RIP} and \mathbf{VIP} are the sets of rIP addresses and vIP addresses respectively. With a single vIP address per host, RSM doesn't need port numbers to identify the different vIP addresses. It follows that the simplex function is one-to-one rather than many-to-one, like FRVM's multiplexing. The RSM controller generates and manages a pool of vIP addresses that it instructs switches to map and forward. The RSM controller's generation of vIP addresses follows Algorithm 6 with random numbers generated with a cryptography secure method using randomness collected by the OS. After an address mutation, the hosts' vIP addresses are allocated from storage, and a new set is generated for the next address mutation and moved into storage. With OpenFlow, the RSM controller instructs switches how to forward and map IP address fields of ARP and IP packets. For transparent mapping to the hosts, RSM routes on the network layer [34] to allow matching via rIP and vIP address fields. Specifically, the modified packets will be either ARP or

IP packets. Similar to FRVM, ARP packets are modified to facilitate network level communication. Although due to hosts having only one vIP address, a queue approach is not needed. A high-level interpretation of RSM is shown in Algorithm 5.

3.7 Measurement Instruments

Measurements were taken on a virtualised network described in Section 3.5 and executed in the test environment specified in Table 3.1. The SDN controllers used for the experiments are described in Section 3.6. Security and performance testing used different methods due to the differing aims, the measurement instruments for each are outlined separately in the following subsections.

3.7.1 Measuring Security

FRVM defends networks with address mutation and multiplexing against the first two links in the attack chain, reconnaissance and access [30]. Nmap [28], an attacker tool, can perform both reconnaissance and access through its host and port discovery. It follows that the level of security of tSDN, RSM, and FRVM controllers was assessed using data logs from Nmap network scans completing the two phases: host and port discovery.

Host discovery involves identifying a host within a network, usually with an IP address. This uses common packets that are likely to receive a response, like ICMP and common transport layer ports. This stage involves reducing the number of potential hosts, to save time in port discovery. Port discovery involves finding detailed information about the identified hosts through a

series of port probes per host. Although it is not necessary to first complete host discovery before port discovery, it dramatically reduces the total amount of probes. Host discovery recorded the number of discovered ports per host.

Port discovery techniques uses transport layer protocols to discover open ports on hosts, there are a multitude of techniques including UDP and many TCP methods. A subset of the port scanning techniques, UDP and TCP connect, SYN, FIN, NULL, and XMAS were chosen to meet the time restrictions of the thesis. The UDP and TCP methods were chosen over other port scanning techniques due to these transport protocols being more common than alternatives. Port discovery recorded the number of open ports per host.

For the scans, Nmap measured the hosts discovered along with their detected ports and the scan duration. The scanning speed of Nmap was decided by its dynamic behaviour, setting a specific speed was avoided as fast scans can be easily detected. Nmap was chosen over alternatives such as Nessus [2] due to its well-documented use, popularity with both attackers and defenders [28], and familiarity with the tool allowing time constraints of the thesis to be met.

3.7.2 Measuring Performance

Performance was measured with the duration of TCP file transfers. Small delays were avoided, like latency caused by OpenFlow mechanisms, due to the small times. For these small delays, it was difficult to discern differences between the SDN controllers due to the virtualised network causing small measurement error that obscured the differences. The server and client were positioned inside and outside of the SDN-based network respectively. The TCP client and server were implemented in Python, recording the transfer duration using the time since the Linux epoch. Timers that record Central

Processing Unit (CPU) time typically have a finer accuracy resolution but do not record blocking calls commonly used in networking and thus were not appropriate. The timing method used had a resolution of approximately 1ms and due to the magnitude of file transfer durations, it was appropriate.

3.8 Data Collection

FRVM is a MTD security technique proposed to defend against network reconnaissance and access. Security and performance testing were conducted on the FRVM controller to demonstrate its capabilities and discover future research directions. Data collection is split into two categories: security and performance testing. In both sections, FRVM was compared with the tSDN and RSM controllers described in Subsections 3.6.2 and 3.6.3, respectively.

3.8.1 Security Testing

The aim of security testing was to demonstrate and verify the defence of FRVM through evaluation alongside software implementations of the tSDN, RSM, and FRVM controllers with a real-world attacker tool, specifically a network scanner Nmap. Despite FRVM currently lacking any considerations for TCP connections, as discussed in Subsection 3.2.3, this did not affect the validity of the security testing data. TCP scanning either doesn't create a connection or creates a very short-lived connection, for the case of the TCP connect scan. The security of FRVM was measured through the success of Nmap scanning on a FRVM protected network relative to the tSDN and RSM networks. The attacker's success was measured by the level of information disclosure and scan duration. Information disclosure is expressed in terms

of the host and port discovery. The duration of the scan directly effects the amount of attacker gathered information that remains valid at the end of the scan.

Host discovery

Host discovery involves identifying a host within a network, usually with an IP address. Although it is not necessary to first complete host discovery before port discovery, it dramatically reduces the total amount of work. Nmap offers many end-to-end host discovery methods including ICMP, ARP, UDP, and various TCP probes. The host discovery method for security testing had to be capable of detection over the Internet, discounting ARP probes. The remaining scans all function over IP and therefore can discover hosts over the Internet. Ultimately, the default options in Nmap were chosen due to their common use and the modelled attacker is unaware of MTD. The defaults include multiple probes: ICMP echo and timestamp request and TCP SYN and ACK packets.

The design of FRVM avoids host discovery through its address multiplexing, as discussed in Subsection 3.2.2. For this reason, host discovery is disabled in scans against FRVM, meaning each network scan begins with port discovery assuming all hosts in the IP subnetwork are active. For analysis of scans on a FRVM deployed network, a host was considered discovered if any number of ports had been discovered on that host.

Port discovery

Security testing with Nmap used TCP and UDP port scanning techniques. Of the TCP scan, five sub-types were tested: TCP connect, SYN, NULL, FIN, and

Xmas [28]. Several different port scanning techniques were used to discover those that posed the largest risk against the FRVM controller. For the MTD controllers, RSM and FRVM, it seemed likely that faster scans would be the most threatening as these would minimise the number of address mutations that could occur during a scan. The most threatening scans were found using an address mutation interval of 300 seconds, as performing all scan types for multiple address mutation intervals was out of the scope of this thesis. After finding the most threatening scans, these were performed on the MTD controllers for a variety of address mutation intervals to discover the effect on information disclosure. The tested address mutation intervals were 30, 90, 180, and 300 seconds.

Scanning Strategies

The duration and information disclosure were recorded for each network scan, taken from the perspective of the attacker. The duration of the scan was simply the time between the attacker beginning and ending the scan. Information disclosure was measured by the number of discovered hosts and open ports. The effect of address mutation and multiplexing was measured with two different scanning strategies: scans run until completion and scans run for a single address mutation interval.

Scans that were run until completion, or full scans, could experience any number of address mutations throughout the scan. This analysed effects on an ongoing scan that was not MTD aware. Specifically, full scans provided the effects of address mutation and multiplexing on extended duration and obfuscation of information disclosure. Full scans begun at any point after an address mutation, this assumed the attacker either was not aware of the MTD protection or unable to predict the movements. Unfortunately, full scans do

not fully describe the information disclosure such as the amount that can be found in a single address mutation interval. This is because the information is obscured between intervals, leading to false numbers of hosts and ports.

Scans that were run for a single address mutation interval, or partial scans, found the level of information disclosure that could be obtained before it expires. This analysed the difference between the MTD controllers due to address multiplexing. Additionally, partial scans analyse the amount of data that could be collected before anything expires. The tSDN controller was excluded from partial scans, as it doesn't apply dynamics and therefore the time restriction does not apply.

Security Testing Configuration

All the SDN controllers were tested on the same virtualised network described in Section 3.5. The network contained three switches, an outside user, a network controller, and a variable number of protected hosts. Each protected host randomly opened 10 ports in the inclusive range of 50000-50999. This range was assumed to contain the 1000 most likely open ports, although for simplicity was consolidated into a single block. For TCP scans, the open ports operated a HyperText Transfer Protocol (HTTP) server, implemented in Python [44], to respond to the connection requests of the TCP scans. For UDP scans the open ports offered an echo service, as Nmap UDP scans detect open ports on any response. The port numbers of offered services were generated with a cryptography secure random number generator. The user outside of the network was modelled as an attacker performing a Nmap scan using its default dynamic behaviour. The attacker scanned over the range 50000-50999 over a class C IP address range. This assumed the attacker knew the subnet size of the network and the port range of randomly

opened ports. The subnet size could be guessed from the subnet of an advertised address using utilities like Whois [7]. The port range was known from a list of well-known ports [28], as the host's ports were within the 1000 most likely open ports. Finally, the router was excluded from the attackers scan as FRVM doesn't protect the networks routers, as described in Subsection 3.6.1.

3.8.2 Performance Testing

Inherently FRVM will cause an additional delay by managing, mutating, and multiplexing between the vIP and rIP address spaces. To better understand FRVM's performance repercussions, its effect was measured with respect to the duration of file transfers over TCP. In the file transfer, protected hosts acted as servers and hosts outside of the SDN network acted as clients. The duration of file transfers was measured from the start of the download until the complete file had been transferred, from the perspective of the client. Address mutations were not considered whilst measuring the file transfer of FRVM because currently there is no solution for connection dropouts between address mutations, as explained in Section 3.2.3.

The overhead of the SDN controllers was further analysed by measuring the average duration of file transfer for multiple servers whilst under network load. The network load was varied with the number of concurrent file transfers. Specifically, two methods were used: increasing the number of servers and increasing the number of file transfers from a server.

All file transfers were measured on the virtualised network described in Section 3.5. The network contained three switches and a variable number of users and protected servers. Both the client and server communicated

through TCP to transfer a 100MB file from the server to the client. Delays associated to resolving a domain name were not counted towards the duration of the transfer.

3.9 Data Analysis

Data was arranged in terms of research problems and interesting properties found in testing. All the data analysis was performed with a statistical computing language, R. The security and performance data were analysed with explanatory analysis and failure time analysis respectively.

3.9.1 Security Analysis

The measuring instrument for security testing was Nmap. Nmap scans output data for each host on the network. To simplify the interpretation of the results, metrics were aggregated per network scan. For example, the number of open ports found per host was computed as an average over a network scan. Due to the dynamic behaviour of Nmap's scanning strategies in conjunction with the effects of address mutation and multiplexing the collected data had high variability. Rather than attempting to fit a model to the variable data, an exploratory approach was taken. This involved describing the shape, centrality, and variability without distributional assumptions. Visualising data is especially important for exploratory statistics, an R package called ggplot [50] was helpful in generating the visual plots used to describe the scan data. Whilst visualising the data, outliers were removed on a per scan type basis using the scan duration metric. The outliers were removed

using the InterQuartile Range (IQR) method, that is samples below

$$Q_1 - \frac{3}{2} \times IQR \quad (3.13)$$

or above

$$Q_3 + \frac{3}{2} \times IQR \quad (3.14)$$

were counted as outliers and removed from the data set, where Q_1 and Q_3 are the lower and upper quartiles, respectively.

3.9.2 Performance Analysis

Performance testing data measured the duration of an event thus failure time analysis could be applied. Specifically, a class of semi-parametric survival models called the Accelerated Failure Time (AFT) models [25]. As AFT models are semi-parametric, the models can vary based on the distribution of their error variable. The distribution of an AFT model effects the data it can fit, thus the most appropriate distribution was chosen through model selection with the Akaike Information Criterion (AIC). It follows that the distribution was chosen by the fitted model that had the minimum AIC value. The AFT models were fitted to the data and analysed using the R packages: survival [48] and flexsurv [20]. AFT models are log-linear; therefore, the proportional difference of the coefficients was found with the following.

$$\% \Delta Y = 100 \times (e^{\beta_i} - 1) \quad (3.15)$$

Where Y , X , and β_i are the response, explanatory, and coefficient variables, respectively. The coefficients are interpreted differently due to the multiplicative way in which the covariates of a log-linear model act on the response variable. The means of different groups were computed from the fitted AFT models and confidence intervals were found through simulations on the normal asymptotic distribution. Hypothesis testing was performed to conclude a difference between the controllers. Firstly, an omnibus method was used to reduce the chance of a Type 1 error, followed by individual pairwise tests.

3.10 Reproducibility of the Experiment

The reproducibility of this thesis is ensured by including the entire repository of source code including Python and R files, raw data, and other related materials. The Python code includes the configuration of the virtualised network, implementation of SDN controllers, and written tests for measuring security and performance. The R code includes the data preparation and analysis of security and performance. There is a large quantity of raw data related to security analysis, in the form of Nmap's Extensible Markup Language (XML) output. The repository is available at <https://drive.google.com/open?id=1q-73XFdXTqmaLG8mK0uh5qbk8pI5T8Ah>, or alternatively the repository could be requested by emailing coledishington@gmail.com.

3.11 Summary

Flexible Random Virtual IP Multiplexing (FRVM) is a Moving Target Defence (MTD) technique that defends against reconnaissance and access through address mutation and multiplexing although the technique remains theoretical.

It follows that all analysis performed has been analytical, without considerations for real-world infrastructure. This thesis performed realistic evaluation of FRVM deployed on a virtualised network alongside real-world protocols. The evaluation was performed with respect to security and performance. For comparison with FRVM, two additional Software Defined Networking (SDN) controllers were implemented: the typical Software Defined Network and Random Simplex vIP Mapper.

Security testing involved an attacker outside of the SDN-based network scanning for hosts and their open ports using a variety of port scanning techniques. The most potent port scanning technique was found with an address mutation interval of 300 seconds. Scans with the most potent port scanning technique were then carried out on a variety of address mutation intervals. The resultant data was aggregated per network and analysed with exploratory statistics.

Performance testing involved measuring the extra delay between a server and client, with different MTD controllers deployed server side. Additionally, the effect of an increased number of servers and concurrent connections to a server were individually analysed to discover how FRVM performed under different levels of network load. The resultant data was analysed with survival analysis, using an accelerated failure time model chosen through model selection using the minimum Akaike Information Criterion value.

Chapter 4

Results and Analysis

The software implementation of Flexible Random Virtual internet protocol Multiplexing (FRVM) was deployed on a virtualised network to obtain security and performance data. Additionally, security and performance data were collected for the typical Software Defined Network (tSDN) and Random Simplex internet protocol Mapper (RSM) controllers, for comparison with FRVM. The security data was collected from network scans of the virtualised networks. The performance data was collected from the duration of file transfers, over the Transmission Control Protocol (TCP), at varying levels of network load.

This chapter will outline the preparation of data and conversion to results. Analysis of the results will demonstrate the security and performance trade-off and validate the pre-existing analytical results [42].

4.1 Security Analysis

FRVM is a Moving Target Defence (MTD) technique devised to protect a network against reconnaissance and access. The defence of FRVM was verified through network scanning where the extra protection gained from FRVM

was judged by scanning and comparing results with the tSDN and RSM controllers. The ability of network scans was judged with two different scanning strategies: full and partial scans. Full scans were run until the scan completed, often experiencing multiple address mutations while scanning the MTD controllers. Whereas partial scans were run for a single mutation interval. The tSDN controller was excluded from all partial scans as it doesn't apply dynamics, therefore, the restriction to a mutation interval doesn't apply. Network scanning offers many port scanning techniques, the most formidable of which was found by comparing scans of tSDN, RSM, and FRVM networks between the different port scanning techniques. After finding the most threatening port scanning technique, scans were conducted against the RSM and FRVM controllers for multiple different mutation intervals to analysis its effect. The following outlines the preparation of scan data, comparison of multiple port scanning techniques, and the effect of different mutation intervals. All analysis of the data was performed through exploratory statistics involving visual displays rather than relying on distributional assumptions and summary metrics. This approach was taken due to the large variability in the data, caused by Nmap's dynamic scanning behaviour in conjunction with address mutation and multiplexing.

4.1.1 Data Preparation

The raw data of the Nmap scans performed by an outside attacker was in the form of Extensible Markup Language (XML) files. This contained data on the scan and discovered hosts such as scan protocol, number of discovered hosts, and the status of their ports. The data was filtered and processed down to the metrics: number of discovered hosts, average number of open ports per host, and scan duration. For scans that disabled host discovery, hosts were

considered discovered if any number of ports had been discovered. As the data was collected per network, the number of open ports per host had to be aggregated. Aggregating the open ports per host involved averaging the total number of open ports discovered over the total number of hosts in the network. Outliers were removed based on the scan duration by using the interquartile range per scan type, as described in Section 3.9. Each scan type sample distribution only contained outliers that were larger, due the address mutation massively increasing the duration of scans. These were removed as they were not representative of the scan types ability.

4.1.2 Comparison of Multiple Port Scanning Techniques

Attackers have a variety of port scanning techniques available for probing inside of networks to learn information to launch an attack. During security analysis, TCP and UDP port scans were performed on networks with each of the SDN controllers deployed. This demonstrated the differences between the port scanning techniques, with respect to information disclosure and duration. The port scanning techniques were compared with a mutation interval of 300 seconds for the RSM and FRVM controllers, this allowed the effect of movement to be discovered whilst dramatically reducing the number of samples. The following covers the full and partial scanning strategies for comparing the multiple port scanning techniques.

Full Scans

The analysis of the full scan results for different scan types are split into host discovery, port discovery, and scan duration. The RSM controller was configured with a mutation interval of 300 seconds. The results of two full scans

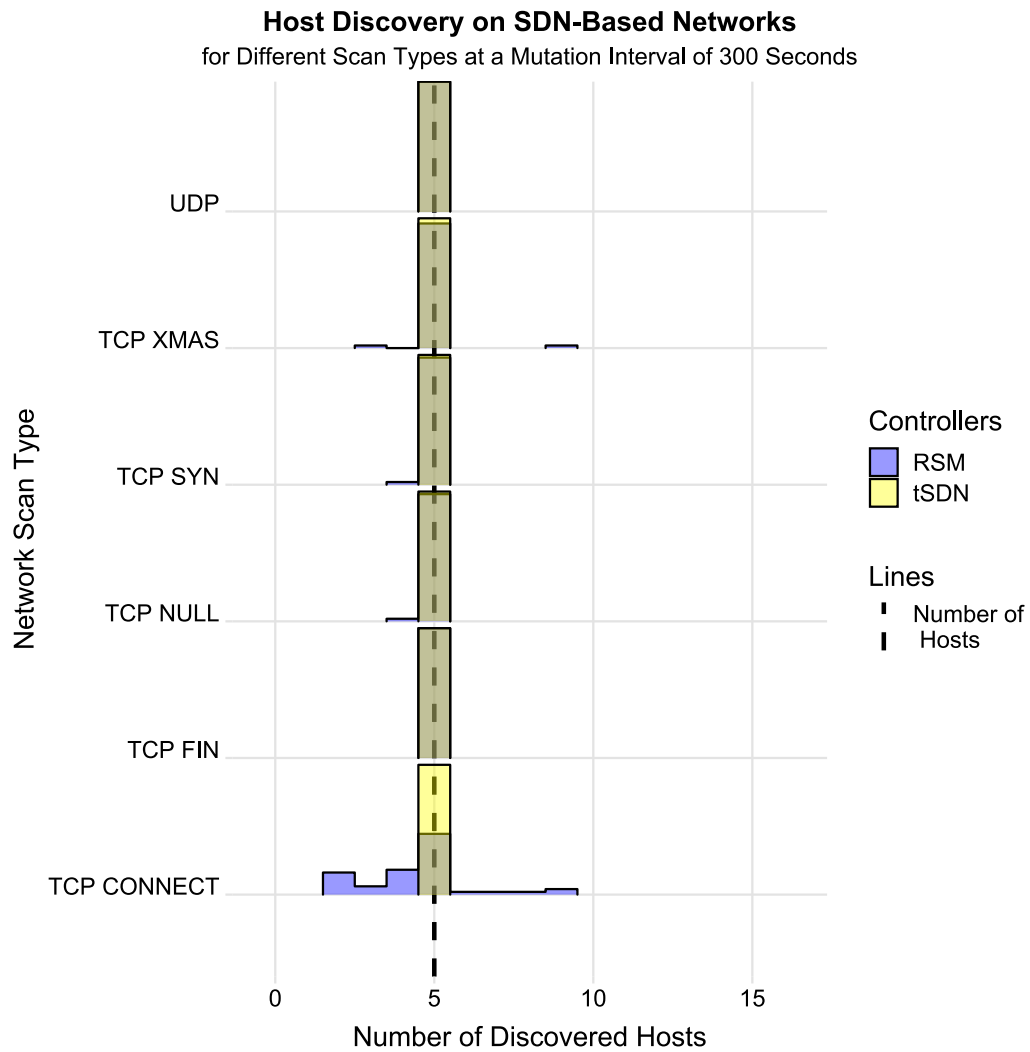


FIGURE 4.1: The sample distributions of the number of discovered hosts from full scans on tSDN and RSM deployed networks, with multiple scan types. RSM was configured with an address mutation of 300 seconds. The dashed line marks the number of hosts in the network.

Scan Type	Duration	Up Hosts	Average Open Ports Per Host
TCP SYN	118351.59	58	12.60
TCP CONNECT	50375.64	35	8.20

TABLE 4.1: Full scans collected from a FRVM deployed network with an address mutation of 300 seconds. Identical to the scans on the tSDN and RSM, there were 5 hosts, each with 10 open ports.

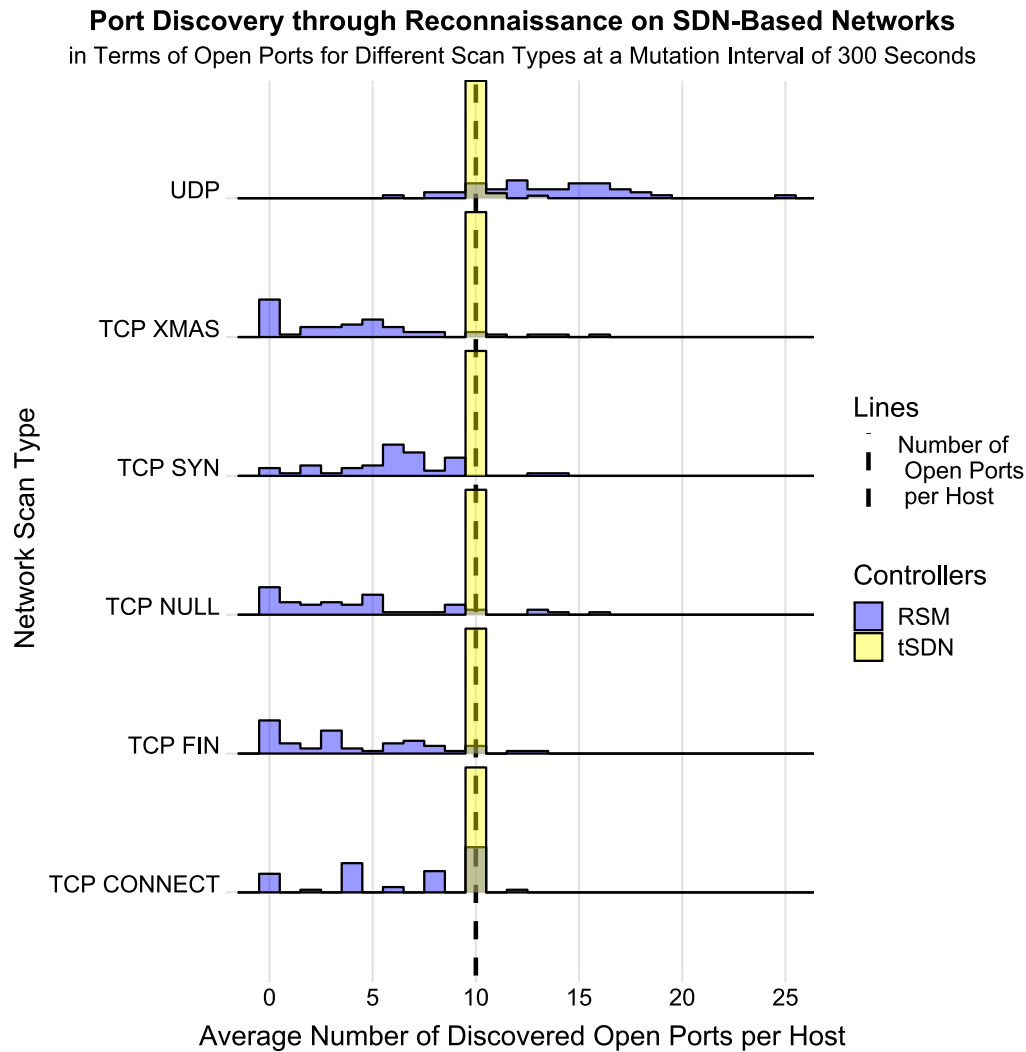
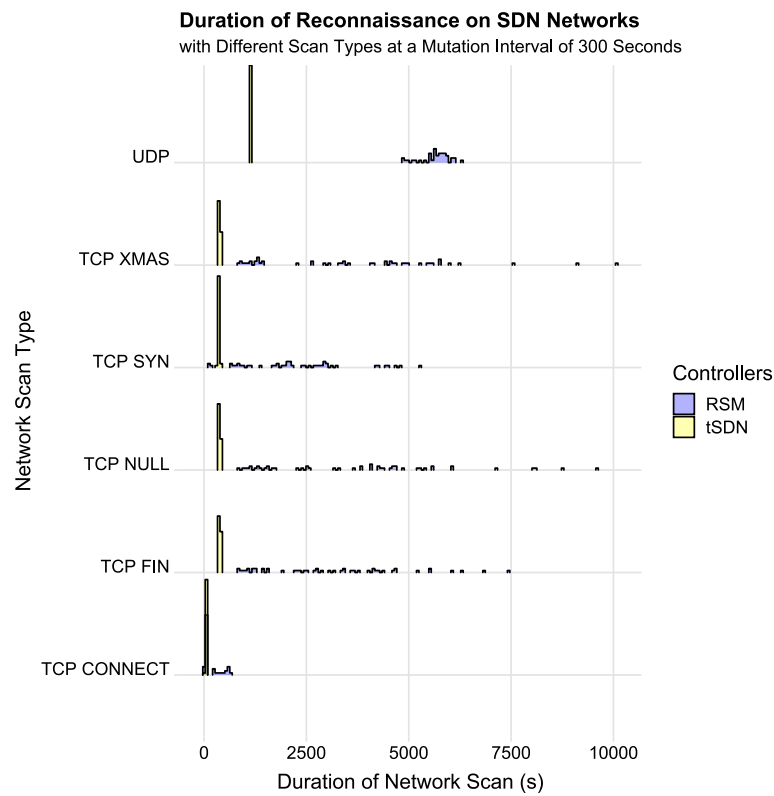
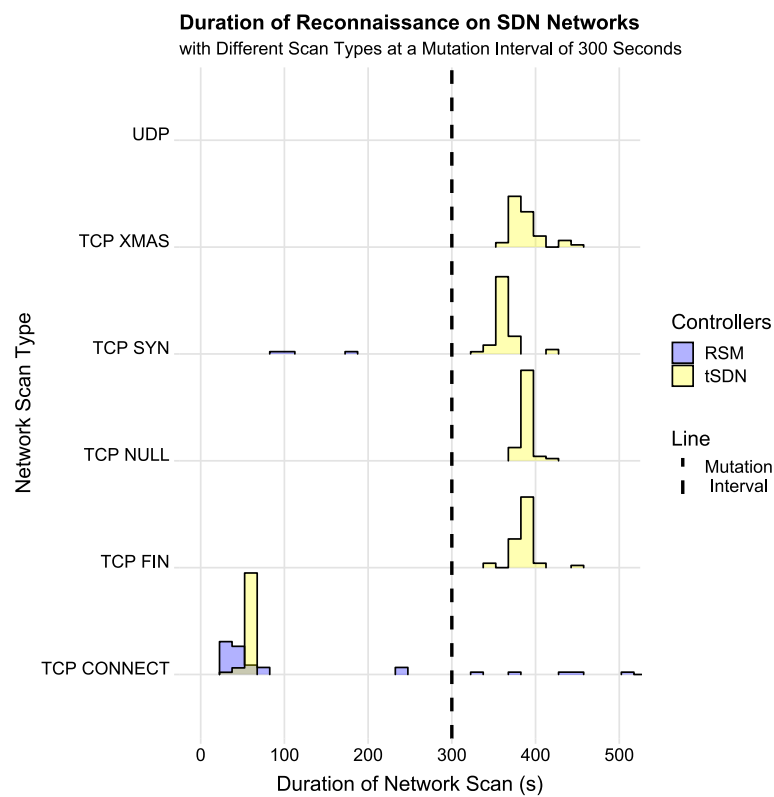


FIGURE 4.2: The sample distributions of the average number of discovered open ports per host discovered from full scans on tSDN and RSM deployed networks, with multiple scan types. RSM was configured with an address mutation of 300 seconds. The dashed line marks the number of open ports per host in the network.



(A) Large range



(B) Shortened range

FIGURE 4.3: The sample distributions of the duration of Nmap scans on tSDN and RSM deployed networks, with multiple scan types. RSM was configured with an address mutation of 300 seconds. The two sub-figures show the same data on different x-axis ranges.

for FRVM are shown in Table 4.1, as can be seen, these scans have a very large duration leading to only single samples. Although the effect of address movement on network scans can be extrapolated from scans against the RSM deployed network.

The sample distributions of host and port discovery for tSDN and RSM deployed networks are shown in Figures 4.1 and 4.2, respectively. Immediately it is clear that the scans against the tSDN network have no variability in the number of hosts or ports discovered, with all the scans finding the maximum number of hosts and ports. Whereas against the RSM network, host discovery for TCP connect is spread and port discovery for all scans are spread. For UDP and TCP FIN, host discovery on the RSM network is constant at 5 discovered hosts. The most variable host discovery scan was TCP connect, with most points between 2 and 5; whereas the other scan types having almost all scans discovering 5 hosts. Port discovery scans against the RSM network are spread in the range of 0 to 24.8, meaning some of scans against the RSM network detected more open ports than there was in the network. This occurred a small number of times for all scan types except UDP, in which the majority of port scans found over 10 open ports. This suggests address mutation produces high obfuscation in UDP scans. Comparing the medians of the scan types, both TCP connect and UDP scans median are closest to the true number of open ports in the network. Although the TCP connect scans are spread below 10 whereas UDP's are spread between 8 and 24.8.

The sample distributions of scan duration for tSDN and RSM deployed networks is shown in Figure 4.3. These two snapshots of the same plot, with different x-axis ranges, show the full variability of the scan durations for RSM and compare large point clusters, respectively. Immediately, it is clear that the scan duration on the RSM network is hugely variable, especially in comparison to the tight clusters of durations produced by scanning the

tSDN network. Additionally, the RSM network scans are strongly positively skewed, indicating that a small portion of scans for each scan type are much more effected than others. The most variable scan durations are the TCP XMAS, TCP NULL, and TCP FIN. This suggests that address mutation is affecting the duration of port scanning techniques. The TCP connect scan durations are clustered, even for the RSM network. Further, the TCP connect scan duration for the RSM network is clustered lower than all other scans types, including those scanning the tSDN network. The TCP connect scan on the RSM network likely has such low variability due to its speed, allowing it to finish with a minimal number of address mutations. In fact, the majority of TCP connect scans on RSM had durations lower than the mutation interval. Focusing on the TCP connect scan, both tSDN and RSM scan durations cluster around the same point with slightly more variance in the RSM scan. This shows that even the TCP connect scan experienced address mutations whilst scanning the RSM network. Further, the slightly increased variability shows that TCP connect, and in fact all the tested port scanning techniques, are affected by address mutation.

Focusing on FRVM, shown in Table 4.1, the two samples had much larger durations than scans on the tSDN and RSM networks. Additionally, the number of open ports discovered was higher in both; due to the large number of address mutations that occurred over the large scan duration shown by the large number of hosts discovered.

Partial Scans

Partial network scans were performed against RSM and FRVM deployed networks to find the magnitude of information disclosure in a single mutation

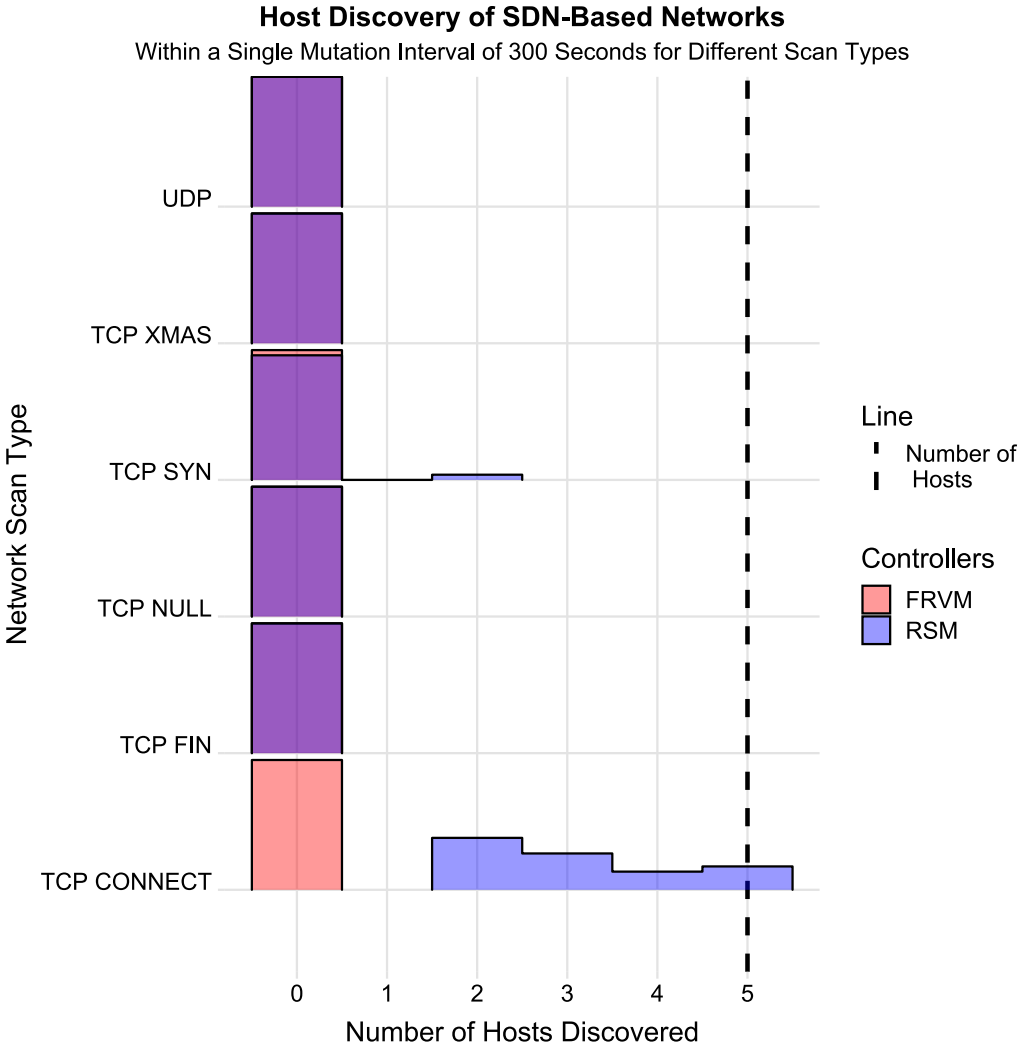


FIGURE 4.4: The sample distributions of the number of discovered hosts within a single address mutation interval of 300 seconds on RSM and FRVM deployed networks, with multiple scan types. The dashed line marks the number of hosts in the network.

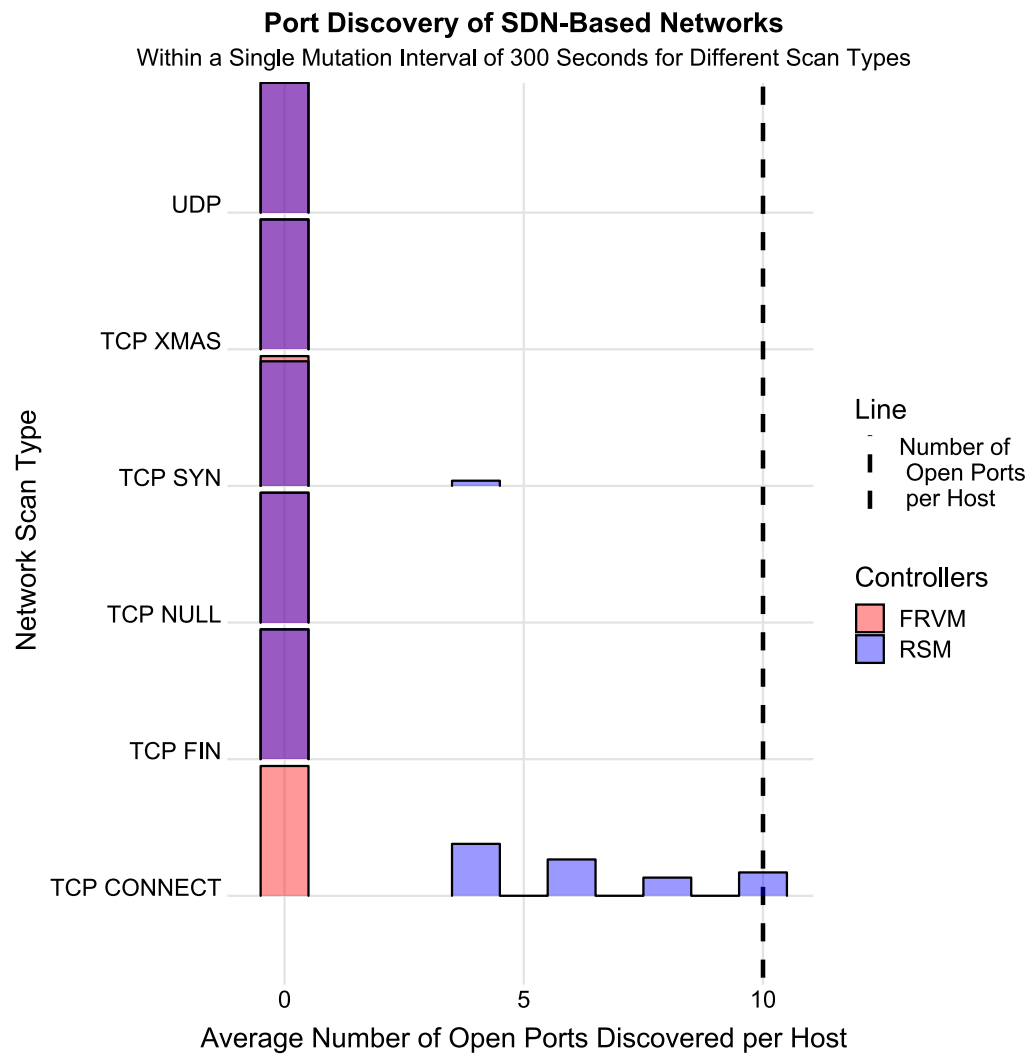


FIGURE 4.5: The sample distributions of the average number of discovered open ports per host; within a single address mutation interval of 300 seconds on RSM and FRVM deployed networks, with multiple scan types. The dashed line marks the number of open ports per host in the network.

interval of 300 seconds, for the different port scanning techniques. The durations were not analysed as these scans were restricted to finish within 300 seconds. Of particular interest was how FRVM's multiplexing of virtual Internet Protocol (vIP) addresses affected the network scans.

The sample distributions of host and port discovery for the RSM and FRVM deployed networks is shown in Figures 4.4 and 4.5. The dashed lines are positioned at the number of hosts and open ports per host in the network. As the host discovery of Nmap was not enabled on FRVM due to its address multiplexing, a host was considered discovered if a single port was found during port discovery. However, no hosts or ports were discovered on the FRVM network. Whereas scanning the RSM deployed network found some success with TCP SYN and connect scans. TCP SYN found 2 hosts in two of its scans and TCP connect found hosts in all scan samples with a positively skewed distribution ranging from 2 to 5. The number of discovered hosts was mostly on the lower end of the range with few scans discovering 5 hosts. Additionally, port scanning is similarly positively skewed, although the clusters of TCP connect are separated by one point due to the aggregation of port scanning results. The skew shows that most of the points are below 10, the true number of open ports per host in the network; although a small amount at the end of its tail find 10 open ports. These results suggest that both RSM and FRVM provide protection against most of the scan types. However, without address multiplexing RSM could not fully protect against TCP connect although it did obfuscate many of the scans. Whereas, FRVM evaded all scans, including TCP SYN and connect due to its increased search space from address multiplexing.

Summary

Full and partial scans were conducted on the SDN controllers, to discover the most threatening scan. This was judged by the level of information disclosure and scan duration against address mutation and multiplexing. The tSDN, RSM, and FRVM controllers corresponded to three different security levels: no protection, address mutation, and the combination of address mutation and multiplexing, respectively. The following outlines a brief outline of how the controllers compare, explanation of UDP scans, and the most formidable scan type.

Without protection, network scans discovered all information in the target network, with highly consistent scan durations. The addition of address mutation caused high variability in the number of open ports discovered by all scans; suggesting that address mutation reduced the possibility of detecting open ports and validity of discovered ports. Further, address mutation caused an increase in scan duration, especially for the longer scan types as more address mutations could occur within the scan. Although TCP connect was not as affected as the other scans due to its speed. The speed of TCP connect, avoided sustaining as many address mutations. Additionally, other scan types have extremely variable and drawn-out durations. Finally, address mutation and multiplexing caused further difficulty, resulting in tremendously large durations for the small number of collected full scan samples. With partial scans, the extra protection afforded by multiplexing avoided even a single open port being detected within a single address mutation interval. Unfortunately, many open ports were discovered on all networks on the full scans. This suggests that further defence should be added to FRVM.

The UDP scans inaccuracy was demonstrated in the full scans against the

RSM controller. The detected number of open ports was consistently higher than the true number of open ports. This revealed UDP's detection method; port probes that do not receive a response mark an open port. This assumption is valid for static systems although address mutation violates this assumption. If an address movement occurs whilst scanning a host, all remaining ports will be detected as open. This obfuscates the discovered information and further slows the scans progress.

The full scans suggest that TCP SYN and connect can discover the most information although TCP connect completes much faster. It follows that after the completion of TCP connect, more of the information would be valid as less address mutations have occurred. This was further demonstrated in the partial scans, with TCP connect discovering far more information than other scans within a single mutation interval. For these reasons, TCP connect was marked as the most threatening scan. Although the results for scanning FRVM did not suggest the most suited scan for an attacker to overcome address multiplexing. However, FRVM uses address mutation as well, therefore TCP connect would threaten FRVM too. With this knowledge, further scans were performed with TCP connect at multiple mutation intervals for the MTD controllers: RSM and FRVM.

4.1.3 The Effect of Different Mutation Intervals

Section 4.1.2 explored the different port scanning techniques and the threat posed by their ability to scan the SDN-based network with different controllers. After comparing the scans, the most threatening was TCP connect as it collected more information due to its relatively quick duration, shown in both scan strategies. Due to the threat of TCP connect, more scans were performed on the network whilst varying the address mutation intervals of

the MTD controllers between 30, 90, 180, and 300 seconds. This captured the effect of smaller mutation intervals on the most formidable scan type. Similar to comparing the scan types, the mutation intervals were analysed using the two different scanning strategies: full and partial scans. The full scans analysed the effect of different mutation intervals on RSM networks. The partial scans analysed how much information could be gained in different sized mutation intervals on RSM and FRVM networks.

Full Scan

Full network scans were conducted on a RSM deployed network, with multiple mutation intervals to analyse their effect on the obfuscation of discovered network information and extension of scan duration. Neither the tSDN or FRVM networks had full scans performed for different MTD intervals. The tSDN controller was excluded as it is static and doesn't apply any dynamics whereas the FRVM controller's scan duration was too large to collect the necessary samples, as seen from Table 4.1. The effect of address mutation on full network scans could be extrapolated from the scans against the RSM network.

Interestingly, the sample distributions for host discovery, shown in Figure 4.6, remain a consistent shape through the mutation intervals 90, 180, and 300 seconds with only small variation and all with the highest peak at 5 hosts. Whereas the size of the peak at 5 hosts is significantly smaller for a mutation interval of 30 seconds. This suggests that the larger mutation intervals were not significantly different in terms of reconnaissance. The port discovery, shown in Figure 4.7, also looks similar between mutation intervals of 90, 180, and 300 seconds, though not to the same degree as host discovery.

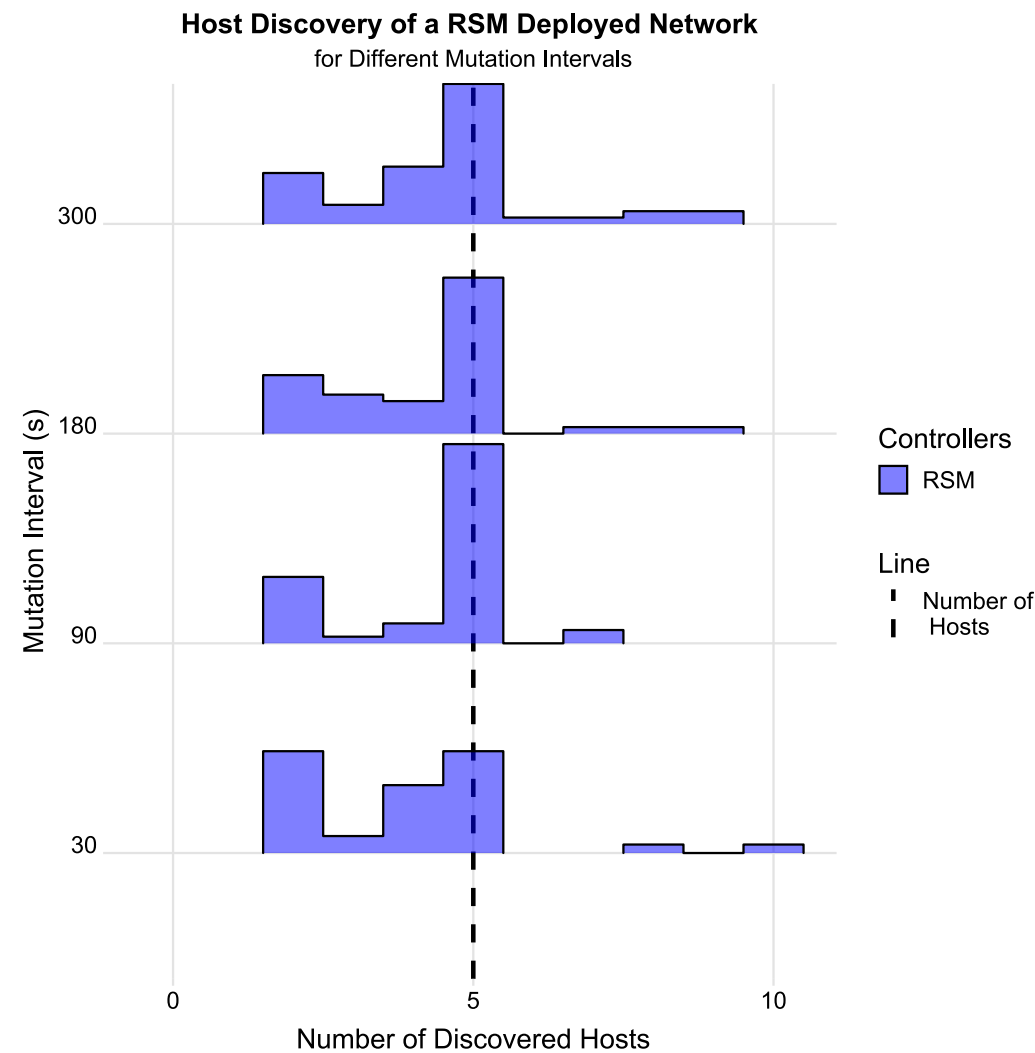


FIGURE 4.6: The sample distributions of the number of discovered hosts, using a full TCP connect scan, for different mutation intervals on a network, with the RSM controller deployed. The dashed line marks the number of hosts in the network.

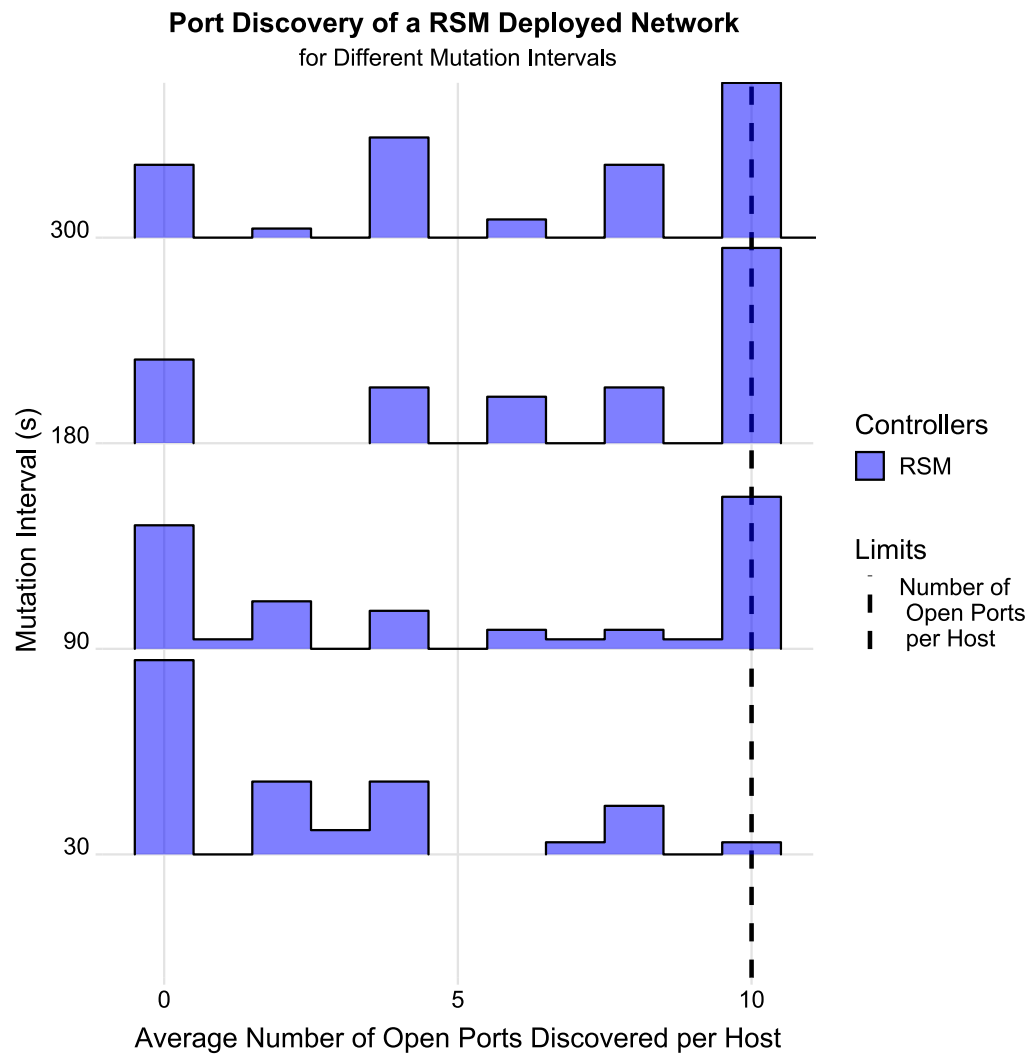


FIGURE 4.7: Sample distribution of the average number of discovered open ports per host, using a full TCP connect scan, for different mutation intervals on a network, with the RSM controller deployed. The dashed line marks the number of open ports per host in the network.

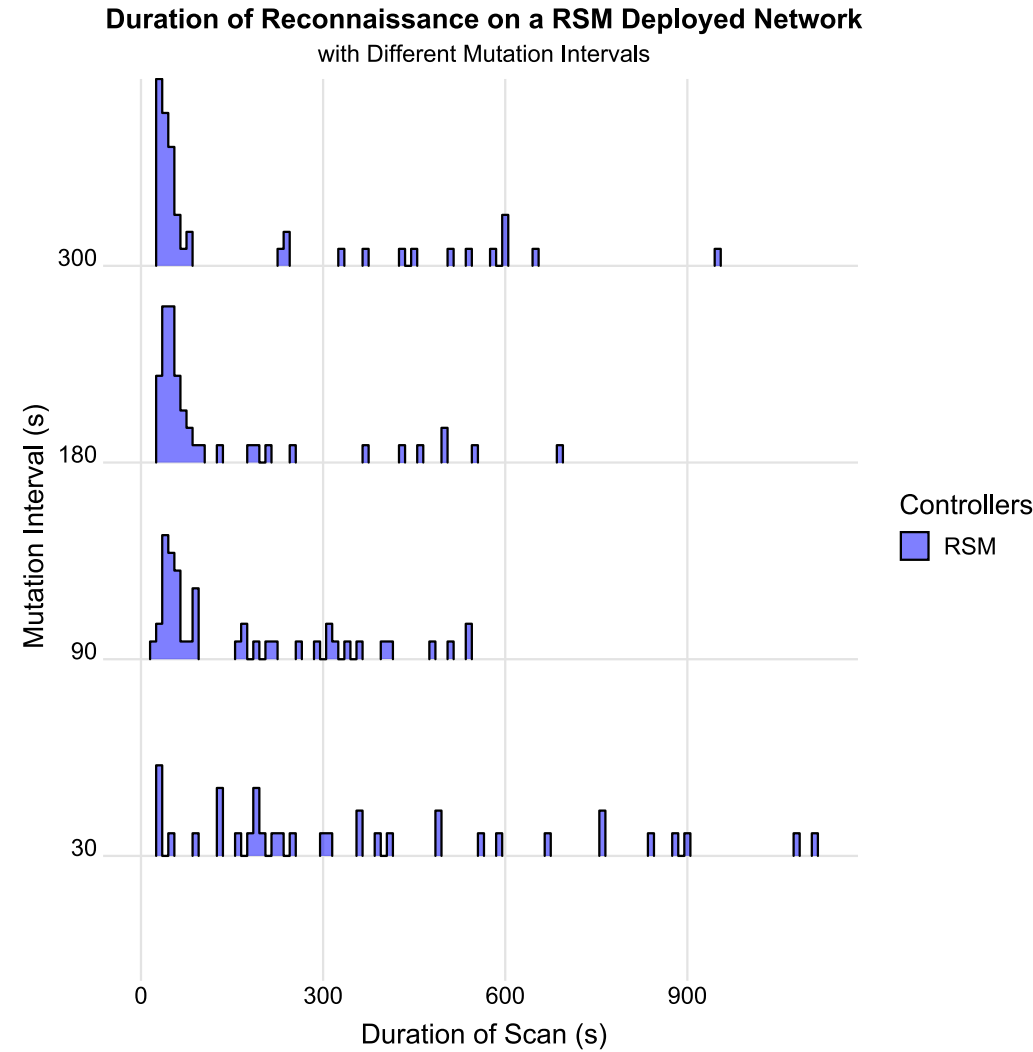


FIGURE 4.8: Sample distribution of Nmap scan durations using the TCP connect scan, for different mutation intervals on a network with the RSM controller deployed.

Similar to host discovery, there is a clear difference between the mutation interval of 30 seconds and the others. This suggests that without a frequent enough mutation interval, the scan is not greatly affected. In both, the scans found less information in most cases for the mutation interval of 30 seconds. Unfortunately, there were still a significant amount of open ports that were discovered on all scans.

The sample distribution for scan duration, shown in Figure 4.8, shows a cluster of scans around 100 seconds with a positive skew that becomes smaller as the mutation interval reduces through the levels. This suggests that increased frequency of the address mutation, results in increased variance in the scan duration. Upon a mutation interval of 30 seconds, there is no longer a large cluster around 100 seconds. Rather, there are small clusters spread below 300 seconds. This shows the increased variability in scan duration with a 30 second mutation interval. Similar to the host and port discovery, the scan duration suggests that the protection relies on an appropriately small mutation interval.

Partial Scan

Partial network scans were performed on MTD deployed networks to analyse how much information could be discovered within a single mutation interval, for different sized intervals. The MTD controllers, RSM and FRVM, were scanned for the different mutation intervals. The tSDN controller was not included as it does not apply dynamics.

The sample distributions of host discovery for the different mutation intervals is shown in Figure 4.9. Scans on the mutation intervals 90, 180, and 300 seconds have all of their samples in the range of 2 to 5 hosts for the RSM network. Both 90 and 180 seconds are very similar and look approximately

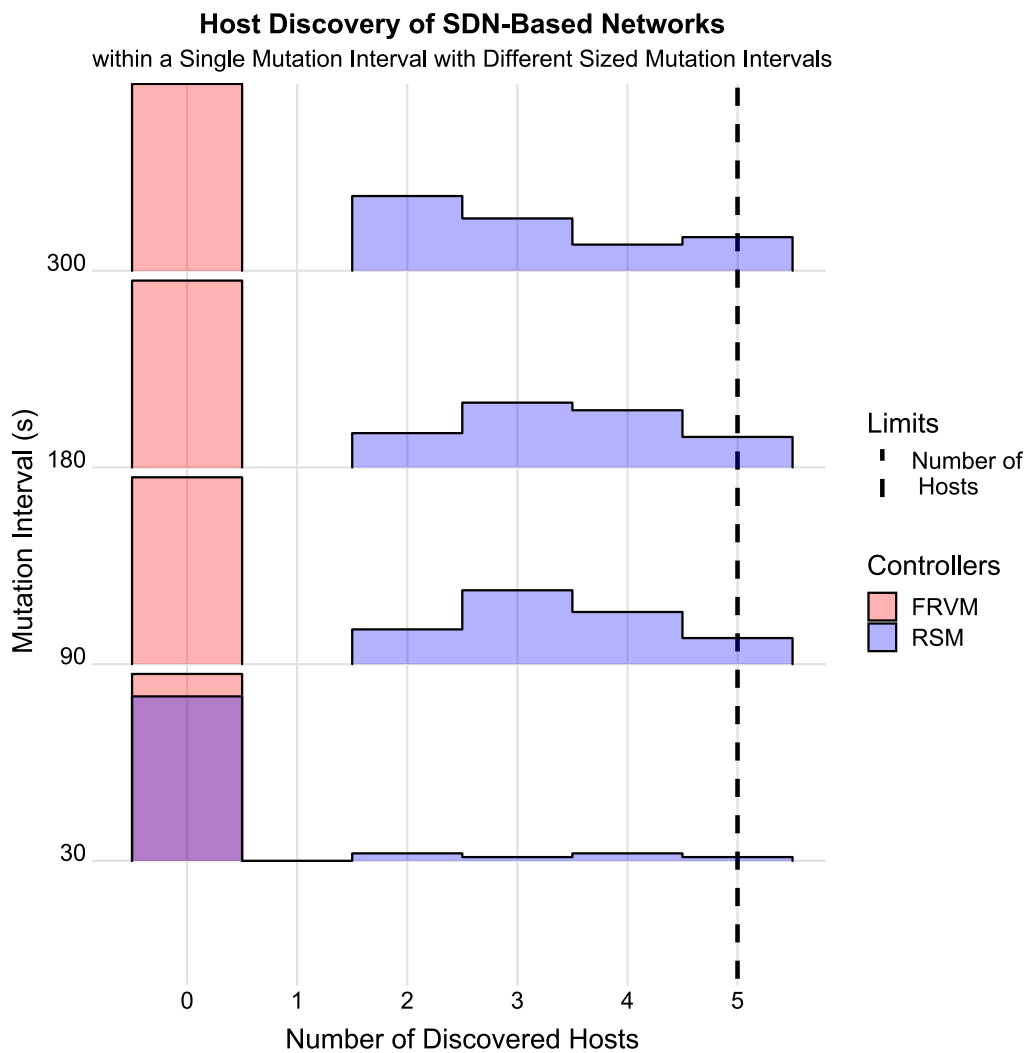


FIGURE 4.9: The sample distributions of the number of discovered hosts within a single mutation interval of 30, 90, 180, and 300 seconds on RSM and FRVM networks. The dashed line marks the number of hosts in the network.

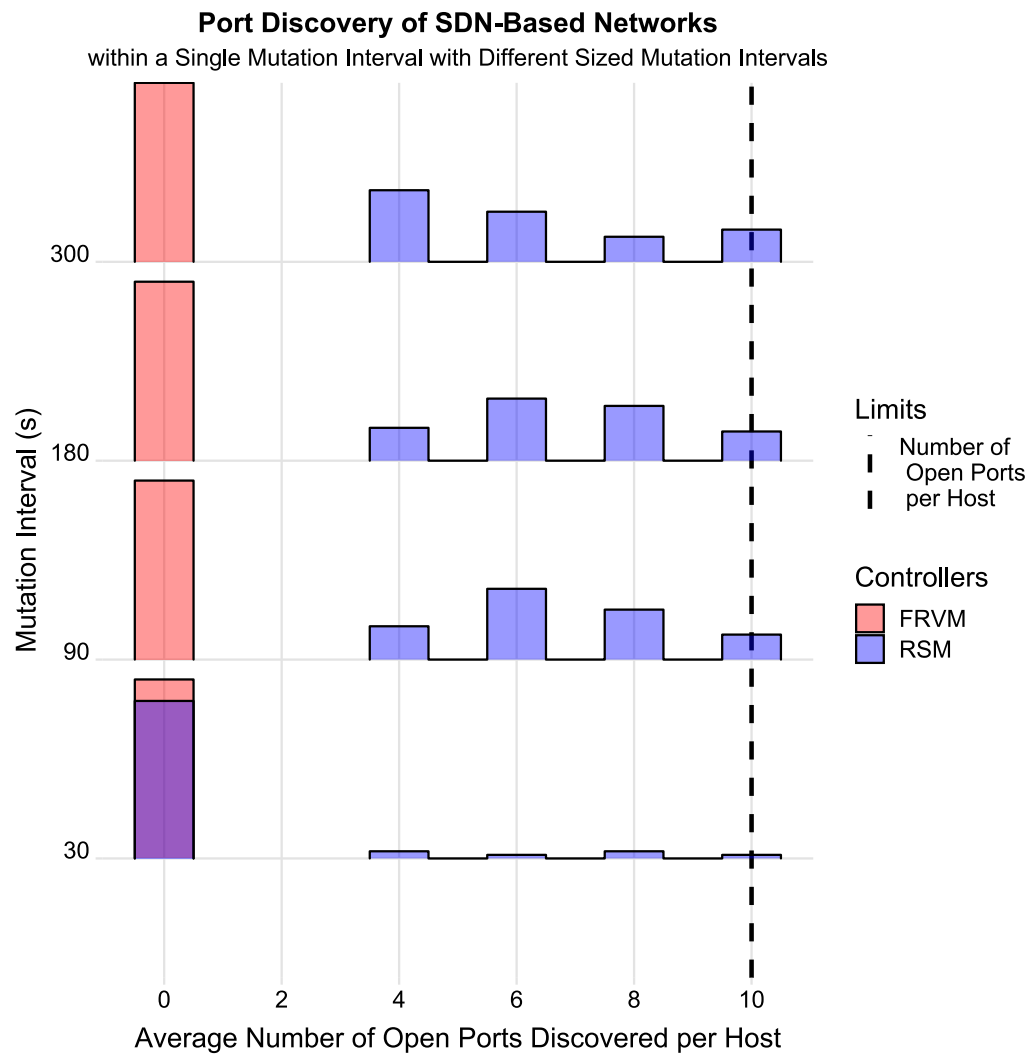


FIGURE 4.10: Sample distributions of the average number of discovered open ports per host using the TCP connect scan within a single MTD interval of 30, 60, 90, 180, and 300 seconds on a network with RSM and FRVM controllers. The dashed line marks the number of open ports per host in the network.

normal whereas 300 seconds is positively skewed. With a mutation interval of 30 seconds, almost none of the hosts were discovered. Sample distributions of port discovery, shown in Figure 4.10, follow the same trend as host discovery, both 90 and 180 seconds appearing very similar with 300 seconds having a positive skew. Although the points are separated into single peaks due to the ports being aggregated. That is, usually if a host is discovered all its ports are discovered. Whereas, almost no open ports were found in scans at a mutation interval of 30 seconds. Almost no hosts were discovered at a mutation interval of 30 seconds, which likely aided this result. Both host and port discovery were completely ineffective against FRVM, finding nothing in all samples.

Summary

Both the full and partial scans suggested that a higher frequency address mutation provided improved protection against network scans. This was shown by the differences between a mutation interval of 30 seconds and mutation intervals of 60, 90, and 180 seconds. Without address multiplexing, address mutation occurring in the host discovery phase significantly reduced the number of ports discovered. This suggested that a reactive approach should be taken to ensure mutation occurs during host discovery. However, this did not apply to address multiplexing, as host discovery is not effective, and all hosts had to be port scanned. There are still many open ports discovered on the RSM deployed network at an address interval of 30 seconds. This suggests that the protection provided by address mutation can be effective but is not enough by itself. FRVM, with the addition of address multiplexing did not have any ports discovered within the partial scans. Again, this shows improved security, but the full scans demonstrated that many open ports can be discovered, even if they are not all valid at the end of the scan.

This suggests that the addition of attacker aware protections to FRVM may be necessary to provide strong protection. However, it seems likely that an IDS could more easily detect scans against the MTD controllers due to their extended durations and increased number of port probes.

4.2 Performance Analysis

FRVM is a security technique with the aim of protecting a SDN-based network from an attacker although performance remains an important consideration. Security techniques that do not have a fair security and performance trade-off are not adopted [47]. The effect of the tSDN, RSM, and FRVM controllers on network performance was measured using the duration of file transfers over TCP at different network loads. Comparing the controllers gave the performance cost of address mutation and multiplexing. Latency and flow management delays were avoided as they could be obscured by measurement error in the virtualised network.

4.2.1 Data Preparation

The duration of the file transfer was recorded by each of the outside users downloading a 100MB file from protected servers. Under load there were multiple durations, measured for each user's connection. The durations of the concurrent connections were aggregated into single metrics by averaging the connection durations by the total number of connections. Whilst measuring file transfer, if any of the concurrent connections failed the data was discounted. The data was discounted as it did not reflect the true network load and the average of the connections could not be computed.

TABLE 4.2: Summary statistics for the explanatory variables of an AFT model fitted to the duration of TCP file transfer.

Variables	Value	Interpretation of coefficient	Std. Error	z value	p value
is_tSDN	-0.2198	-19.73	0.008552	-25.71	<0.0001
is_RSM	-0.01221	-1.214	0.008552	-1.427	0.1534
Number of servers	0.1027	10.81	0.001288	79.44	<0.0001
Ports per server	0.04975	5.101	0.001288	38.64	<0.0001

TABLE 4.3: ANOVA chi-square tests to test the significance of different SDN controller groups for the duration of TCP file transfers at different levels of network load.

Terms	Residence Df	Deviance	p-value
is_RSM + is_tSDN	1344	NA	NA
Base	1346	-649.9	<0.0001
is_RSM + is_tSDN	1344	NA	NA
is_tSDN	1345	-2.037	0.1535
is_tSDN	1345	NA	NA
Base	1346	-647.9	<0.0001

4.2.2 Data Analysis

The file transfer measurements were taken on the virtualised network described in Section 3.5. For the transfer, hosts inside of the network acted as servers and outside hosts acted as clients. The bandwidth of simulated links was restricted to 100Mb/s and 10Mb/s for links within the network and links connecting to the network, respectively. Measurements were taken from tSDN, RSM, and FRVM deployed networks to discover and analyse the transfer delays and how they compared. Additionally, the effect of separately increasing the number of concurrent servers and concurrent connections to a server was explored to discover how well the SDN controllers could handle increased load. Firstly, an Accelerated Failure Time (AFT) model was

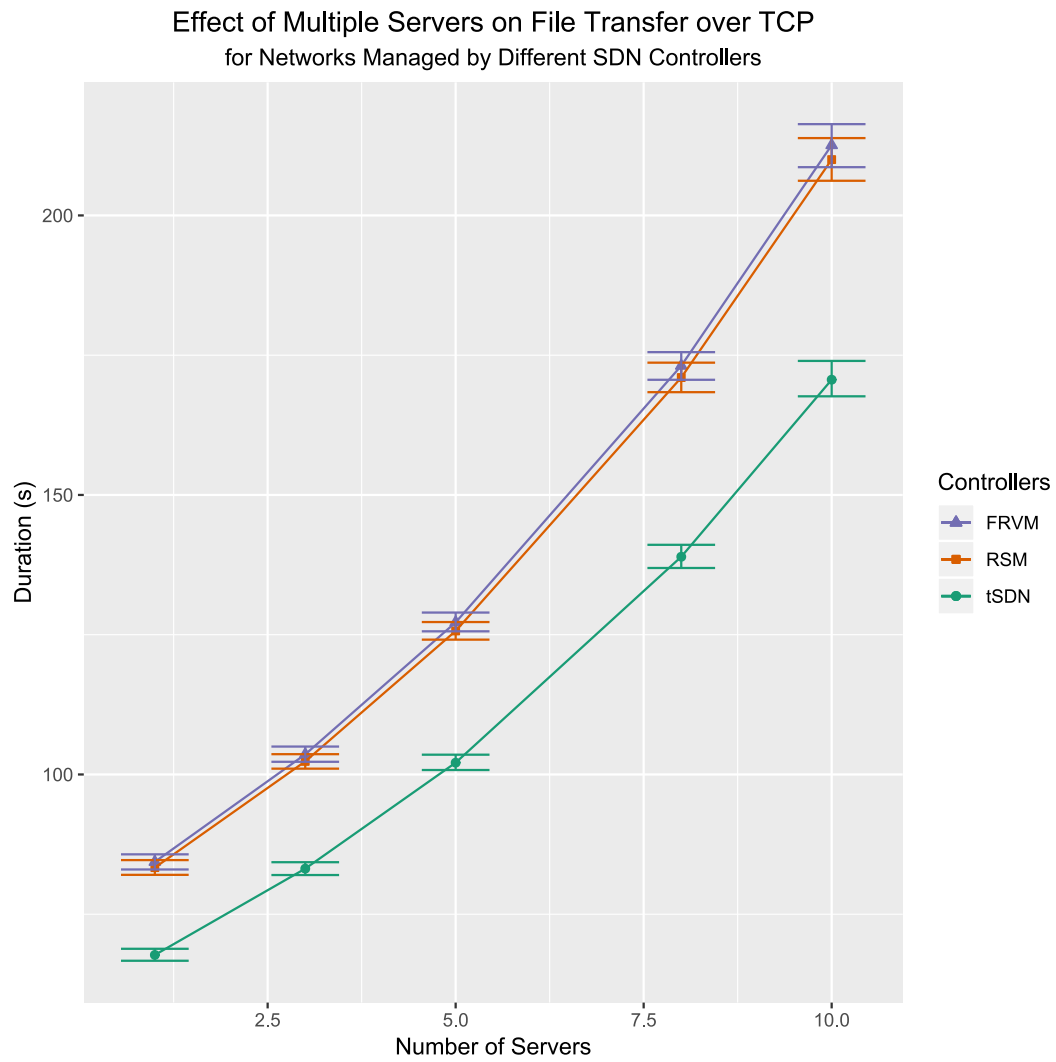


FIGURE 4.11: A line plot of the duration of TCP file transfer against the number of concurrent connections to different servers. The points are means generated from the fitted AFT model in Table 4.2, with the number of connections per server held constant at one. The error bars shown are 95% confidence intervals generated through simulations on the normal asymptotic distribution.

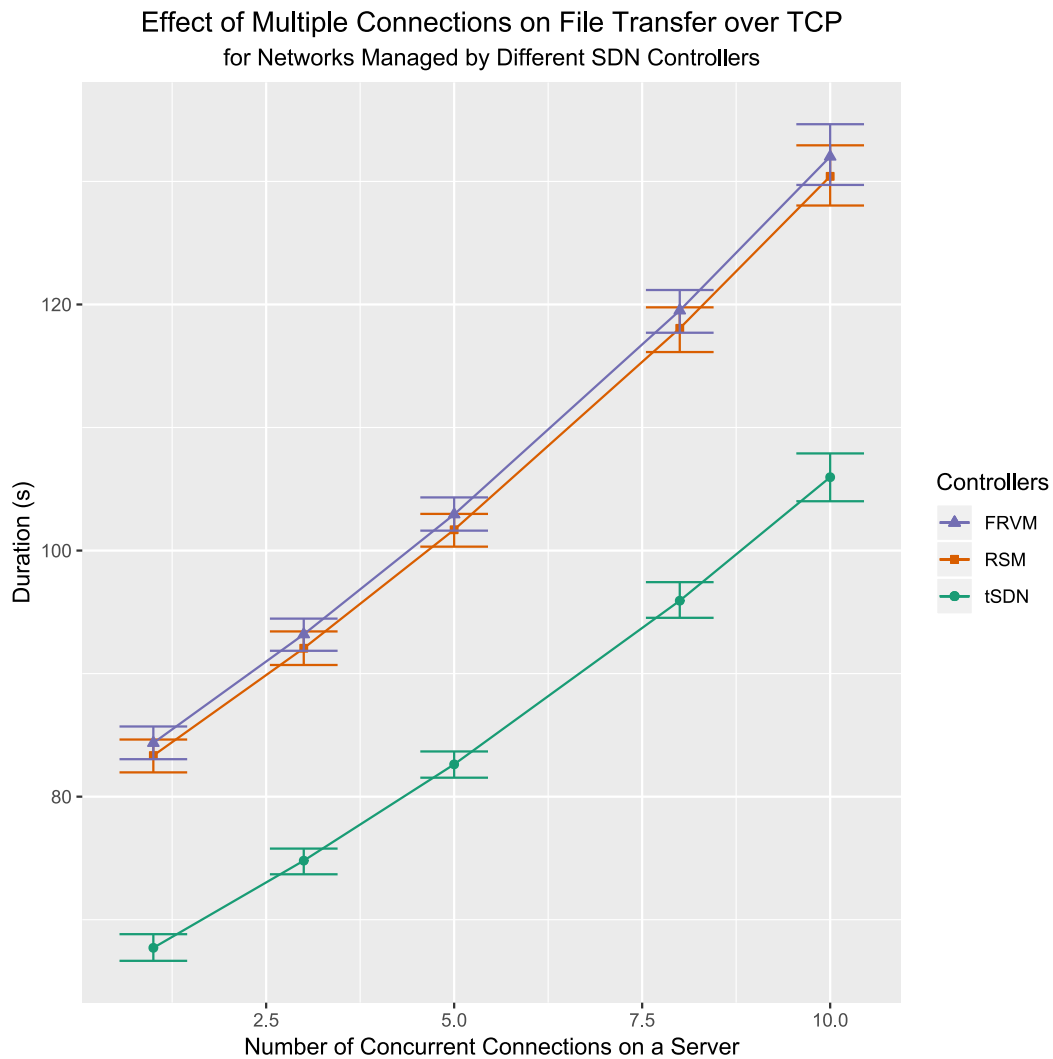


FIGURE 4.12: A line plot of the duration of TCP file transfer against the number of concurrent connections to a single server. The points are means generated from the AFT model in Table 4.2, with the number of servers held constant. The error bars shown are 95% confidence intervals generated through simulations on the normal asymptotic distribution.

fit to the data using a log-normal distribution chosen through model selection using Akaike Information Criterion (AIC) to judge the models. Selection was performed over weibull, exponential, gaussian, logistic, log-normal, and log-logistic models. The fitted model identified the controller groups with dummy variables. The coding

Controller	is_tSDN	is_RSM
tSDN	1	0
RSM	0	1
FRVM	0	0

(4.1)

resulted in two dummy variables: is_tSDN and is_RSM. These modelled the tSDN and RSM with FRVM as a reference. A summary of the fitted model is shown in Table 4.2. AFT models are log-linear and thus the coefficients needed to be transformed to find the proportion difference using the equation

$$\% \Delta Y = 100 \times (e^{\beta_i} - 1) \quad (4.2)$$

where Y , and β_i are the response variable and coefficient of the i^{th} explanatory variable in the fitted model, respectively. The proportional difference is the percentage increase of the transfer duration. It follows that the fitted model found that the download duration of tSDN and RSM deployed networks reduced by 19.73% and 1.214% compared to the FRVM protected network. This suggests the similarities, the address mutation, between the RSM and FRVM controllers causes most of the extra delay on top of simple Ethernet-level routing. Whereas the differences, being address multiplexing, between the MTD controllers resulted in a minimal amount of extra delay. This suggests that multiplexing vIP addresses does not cause a significant

difference in file transfer duration.

The controllers were tested under load by separately increasing the number of servers and connections to a server, the results are shown in Figures 4.11 and 4.12, respectively. The number of servers increased the download duration for all controllers, although the MTD controllers had a steeper increase that become more pronounced as the number of servers become higher. For all points, the MTD controllers had larger transfer durations, clearly indicating the performance trade-off. The transfer durations of the RSM and FRVM networks increased at a similar rate with their points placed within each other's confidence intervals. The effect of multiple concurrent connections to a single server did not increase in the same way, the increase appeared linear. Although again the MTD controllers had very similar durations that were consistently larger than the tSDN transfer duration. The effect was likely less pronounced due to the single server slowing the rate of the connections, it follows that the forwarding was not the bottleneck but rather the server could not keep up with demand.

4.2.3 Hypothesis Testing

For testing the difference between the groups, likelihood ratio tests were used to test the hypotheses

$$H_0: \text{The simpler model fits} \quad (4.3)$$

$$H_A: \text{The simpler model does not fit} \quad (4.4)$$

where H_0 and H_A are the null and alternative hypotheses, respectively. The results of the likelihood ratio tests are summarised in Table 4.3. These tests involve a complex and simple model. The simple model contains a subset of

the explanatory variables contained in the complex model. If the null hypothesis is rejected, the complex model fits and therefore the additional variables are significant. The level of significance for accepting or rejecting the null hypothesis will be 5%. The following models will use the variables x_{tSDN} , x_{RSM} , x_s , x_p , and y_D that correspond to is_tSDN, is_RSM, number of servers, ports per service, and duration of file transfer. Suppose the complicated model

$$y_D \sim x_s + x_p + x_{tSDN} + x_{RSM} \quad (4.5)$$

and the simpler model

$$y_D \sim x_s + x_p \quad (4.6)$$

in a likelihood ratio test. The table shows the hypothesis test has a p-value less than 0.001, this is very significant and therefore the null hypothesis can be rejected. It follows that together the dummy variables identifying the controllers are significant in the model. That is, the controllers have significantly different file transfer durations. Although this test is an omni-bus method, therefore pairwise tests must be conducted to find which variables are significantly different.

Suppose the complicated model

$$y_D \sim x_s + x_p + x_{tSDN} + x_{RSM} \quad (4.7)$$

and the simpler model

$$y_D \sim x_s + x_p + x_{tSDN} \quad (4.8)$$

in a likelihood ratio test. The table shows the hypothesis test has a p-value of 0.1535, this is not significant and therefore the null hypothesis cannot be rejected. It follows that the simpler model is not significantly worse than the complex model, meaning the MTD controllers, RSM and FRVM, file transfers

durations are not significantly different.

Suppose the complicated model

$$y_D \sim x_s + x_p + x_{tSDN} \quad (4.9)$$

and the simpler model

$$y_D \sim x_s + x_p \quad (4.10)$$

in a likelihood ratio test. The table shows the hypothesis test has a p-value less than 0.001, this is significant and therefore the null hypothesis can be rejected. It follows that the simpler model is significantly worse than the complex model, meaning that the additional performance cost from the tSDN to the MTD controllers is significant.

In summary, the duration of a file transfer is larger for networks where the MTD controllers, RSM and FRVM, are deployed, with the effect increasing upon larger network load. For a 100MB file, the difference between the MTD controllers is not significant. These results provide evidence of the performance loss of the RSM and FRVM controllers in comparison to simpler SDN controllers. The gathered results did not consider mutation of addresses, rather the overhead of managing, matching, and modifying the VIP addresses. After a solution for address mutation considering TCP connections has been devised, the overhead of address mutation would be expected to increase. Lastly, the FRVM controller does not create significant performance loss compared to the RSM controller.

4.3 Discussion

FRVM is a MTD technique designed to prevent attacker reconnaissance and access through address mutation in conjunction with address multiplexing. The performance and security trade-off of FRVM was realistically evaluated to validate the pre-existing analytical results [42]. The trade-off was obtained by implementing FRVM and comparison controllers, and deploying them on a virtualised network alongside real-world protocols. Whilst conducting the experiment, this thesis aimed to answer three questions: what extra considerations will need to be handled to implement and deploy FRVM? How does FRVM affect security scanning in terms of information disclosure and scan duration? What is the effect of FRVM on a network's performance? The following are addressed considering the results.

4.3.1 What Extra Considerations Will Need to Be Handled to Implement and Deploy FRVM?

For implementing and deploying FRVM alongside common network infrastructure, extra considerations had to be made such as address mutation on routers and address multiplexing over portless protocols like ARP. Address mutation was not applied to the router to avoid hosts losing their default routes. This change was considered minor, as routers have interfaces on multiple networks and therefore would have non-randomised addresses on attached networks. Address multiplexing was an additional protection of FRVM to improve on past MTD techniques although it assumes the presence of ports on all packets between hosts. Unfortunately, this is not the case and ARP, an essential protocol for communication, is one such protocol. This involved implementing a queue-based approach, identifying the vIP address

that should be used based on the oldest ARP request that had not received a response. This was not necessary in past address masking techniques [49, 41, 23, 22, 21] as there was only a single address per host. The trade-off of FRVM is having a more complicated mapping function for a higher level of security, the latter was shown through network scan results.

4.3.2 How Does FRVM Affect Security Scanning in Terms of Information Disclosure and Scan Duration?

FRVM aims to protect against reconnaissance and access through address mutation and multiplexing. The level of protection was judged by the amount of information disclosure and duration of the scan. The information disclosure was measured by the amount of host and port information gained through reconnaissance and access with a network scanning tool. The duration of the scan was included as a longer scan would have experienced more address mutations resulting in less current information.

Analysis of security results found that address mutation obfuscated and hid the network information, in addition to, dramatically increasing the duration of scans. With address multiplexing, the duration of scans was further increased, resulting in more address mutations occurring throughout a scan.

Multiple scan types were trialed, finding that the TCP connect scan posed the greatest threat against MTD using address mutation. However, with the addition of address multiplexing all scans performed poorly due to a large increase in scan duration. Whilst testing multiple scans, UDP was revealed as a very inaccurate scan against address mutation due to its port detection method. Specifically, UDP scans detect an open port on no response, although all probes elicit no response after an address mutation. Further,

multiple different address mutation intervals showed that better protection is afforded by controllers configured with higher frequency address mutation; this agreed with past research into address mutation techniques [41, 22]. Although the effect of address mutation on scan duration was not gradual, without a significantly small interval the results were very similar. This effect appeared to be both due to the frequency of movement and the movement occurring during host discovery. Although with address multiplexing, host discovery could not be conducted and therefore doesn't need to target mutation towards host discovery.

Unfortunately, many ports were discovered through full scans. Although most would have been expired and not usable for storage-based attacks like hitlist worms; the attack could be launched immediately, or shortly after, the discovery of a port. The benefit of FRVM, and thus address multiplexing, was demonstrated by the large scan durations and evasion of detection within single mutation intervals.

The past analytical research on FRVM [42] measured attack success probability against discovered hosts and number of scans where attack success involves finding a host in the scan. That is, it considers a host exploited if discovered rather than a port of the host discovered. These results and past analytical results agree that FRVM provides better protection than a static network, like the tSDN deployed network.

4.3.3 What Is the Effect of FRVM on a Network's Performance?

The effect of the FRVM controller on network performance was analysed through the duration of TCP file transfer between servers, in the SDN-based network, and outside clients. The effect was judged through comparison

with the typical Software Defined Network (tSDN) and Random Simplex vIP Mapper (RSM) controllers, described in Section 3.6. These tests showed that there was a significant performance difference between Ethernet-level forwarding, used by the tSDN controller, and the MTD approaches, used by the RSM and FRVM controllers. That is, the additional load of address mutation is significant. Interestingly, a significant difference was not found between the MTD controllers thus the addition of address multiplexing, on top of address mutation, doesn't cause a large difference to performance.

The analysis on performance found that address mutation increased the duration of file transfers over TCP with the effect increasing upon larger network load. This suggests a possible scalability issue and future research should be conducted into optimising the address mutation.

Unfortunately, the analytical results composed for FRVM do not include performance measures, although once created they can be compared with these results. These results suggest that, with the security improvements, FRVM would be preferred over similar techniques lacking address multiplexing as the performance loss was shown to be insignificant.

4.3.4 Implications

The contributions of this thesis have implications in professional development and future product development. The results show that the security increase was greater than the performance loss of adding address multiplexing. This could drive future research to include address multiplexing to improve security and anonymity, especially after moving to IPv6 where extra addresses will be abundant. With the lack of realistic evaluation in address

masking MTD, the methodology could aid future researchers through evaluating past or present techniques. Additionally, future research into adaptations of FRVM on a realistic network could be aided through the methodology and codebase. With the rise of techniques like FRVM, future attackers would need to train in MTD and design more complicated MTD aware reconnaissance and access tools. Although due to the many configurable parameters of FRVM allowing deployed networks to act differently, novice attackers may struggle to use the more complex tools resulting in less attacks. Additionally, address mutations highly obfuscate UDP scans, which could lead to a stop on access of UDP ports. The propagation of worms could be significantly slowed due to the large scanning durations. Hitlist worms would also be unlikely to spread effectively as the time to collect network information is far greater than the longevity of FRVM protected host information. Finally, with these results, and the pre-existing analytical results, showcasing the effectiveness of FRVM, the existing networks could adopt the mechanisms with improved trust.

4.4 Limitations of the Study

- The effect of random network disturbances such as queuing delays and packet loss were not measured on FRVM.
- The implementation of FRVM used a simple random key generation method that lacked performance optimisations and attacker awareness.
- Security and performance metrics were gathered on a virtualised network, rather than a physical network. A physical network would produce more reliable results and allow the measurement of smaller delays, like flow modification and latency.

- Matching results for the improvement of FRVM's security with port randomisation were not obtained. These would provide evidence for a change to FRVM's specification.
- Security analysis did not consider the threat from exploits that spread by spamming a particular port of any host. This attack doesn't involve the same level of reconnaissance, access, or development as the exploit has already been chosen.
- Security analysis did not consider the effect of varying the number of protected servers. Although open ports would still have been found, the scan duration would be expected to increase.
- Neither security or performance analysis considered the addition of multiple SDN controllers. Adding multiple SDN controllers would likely improve the performance under network load. It could also limit susceptibility to Denial of Service (DoS) attacks by removing the centrality of the SDN controller [40, 39, 5].
- Nmap [28] assumes a traditional network with static security. With extra considerations for MTD, the ability of an attacker to scan and learn information could be improved.
- The chosen mutation intervals did not showcase when FRVM would become exploitable, with respect to mutation intervals. This was worsened by FRVM's exclusion from full scans due to their large duration and a limited period of time.

4.4.1 Further Research

After conducting the evaluation of FRVM, the methodology and results suggest paths for future research. Firstly, an agenda for further research is outlined to suggest priorities for the work. Secondly, the remaining research directions are listed to aid in additional research directions.

Agenda for Further Research

Further research into the security and performance trade-off of FRVM would involve porting the FRVM controller to a physical SDN-based network to both refresh and obtain new results from a more realistic network setting. All three controllers were written with Ryu, a network operating system, that is used in real networks, therefore a large portion of the thesis code base would be reusable. This research direction would involve configuring a physical SDN-based network, rewriting a subset of the performance tests that were written with Mininet [26], and collecting the performance and security results. There were many performance tests written for FRVM including those to collect small delays, these should help to extend the performance testing of FRVM. This would lead to more realistic results, more considerations for real-world infrastructure, and the ability to test smaller delays.

Further Research Directions

- Demonstrate the disadvantage of a static mutation interval with Nmap. Nmap offers an API, written in Lua [19], that could show how an attacker could discover a static address mutation interval. With host discovery, an attacker could track address mutations until a desired level of confidence is reached.

-
- Explore additional changes to FRVM and test the security improvement against the results obtained in this thesis. Changes could include adding port randomisation, random address mutation intervals, and different methods to retain ongoing TCP connections.
 - The key generation of FRVM should be improved with respect to speed and attacker awareness. Potentially, this would improve scalability and security of FRVM.
 - Explore the overhead related to the Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP) due to the address mutation and multiplexing.

Chapter 5

Conclusion

With the increase of societal reliance and threats against digital infrastructure, the importance of security has increased; however, traditional protections are static causing defenders an asymmetric disadvantage against attackers. Moving Target Defence (MTD) can eliminate this asymmetric disadvantage by adding dynamics to protections. Flexible Random Virtual internet protocol Multiplexing (FRVM) is an MTD technique designed to prevent attacker reconnaissance and access through address mutation in conjunction with address multiplexing. The security and performance trade-off of FRVM was realistically evaluated on a virtualised network. As FRVM was deployed alongside real-world protocols, issues with existing network infrastructure were discovered.

The past literature detailed the numerous issues that adding dynamics to a network can cause and revealed the lacking security and performance analysis of past address masking techniques, in terms of either statistical rigour or realistic evaluation. With the knowledge that FRVM was theoretical, with only analytical analysis, it seemed likely that there would be some issues with existing network infrastructure. Additionally, analysis of FRVM on a virtualised network would improve understanding of the technique, and

future network-based MTD. This led to the following research questions: what extra considerations will need to be handled to implement and deploy FRVM? How does FRVM affect security scanning in terms of information disclosure and scan duration? What is the effect of FRVM on a network's performance?

Whilst implementing and deploying FRVM, issues with a portless network protocol, Address Resolution Protocol (ARP), and router address assignment were discovered and solved. Data was collected and analysed for the security and performance of FRVM. Security testing involved scanning a SDN-based network with Nmap, separate scans were conducted for each of the SDN controllers. Many scans were taken to compare different scan types and find the effect of different address mutation intervals. The results suggested that address mutation was effective at obfuscating and prolonging network scans although open ports could still be discovered without more protection. Further, address multiplexing was effective at further obfuscating and prolonging the network scans. Performance testing involved measuring the duration of file transfers over the Transmission Control Protocol (TCP), at different levels of network load. The results showed there is a significant performance trade-off for mutating addresses although the additional cost of address multiplexing is insignificant. The effect of many individual concurrent connections increases more for larger numbers of servers, suggesting a possible scalability issue with the FRVM controller.

The thesis contributed realistic evaluation of the security and performance of FRVM, along with the detection and solution of issues with existing network infrastructure. The performance cost of address multiplexing on top of address mutation is not significant whereas the additional security is greater, this could lead future research to include address multiplexing, improving security and anonymity. FRVM would force attackers to design complicated

MTD-aware reconnaissance and access tools. Additionally, the comprehensive methodology could aid future research in evaluating MTD protections against reconnaissance and access. Finally, further work could be completed on deploying FRVM on a physical network, to refresh results and collect smaller delays that were avoided in this thesis.

References

- [1] Spyros Antonatos, Periklis Akritidis, Evangelos P. Markatos, and Kostas G. Anagnostakis. “Defending against hitlist worms using network address space randomization”. In: *Computer Networks* 51.12 (2007), pp. 3471–3490.
- [2] Jay Beale, Renaud Deraison, Haroon Meer, Roelof Temmingh, and Charl Van Der Walt. *Nessus Network Auditing*. Syngress Publishing, 2004. ISBN: 1931836086.
- [3] Gui-lin Cai, Bao-sheng Wang, Wei Hu, and Tian-zuo Wang. “Moving target defense: state of the art and characteristics”. In: *Frontiers of Information Technology & Electronic Engineering* 17.11 (Nov. 2016), pp. 1122–1153. ISSN: 2095-9230. DOI: [10.1631/FITEE.1601321](https://doi.org/10.1631/FITEE.1601321). URL: <https://doi.org/10.1631/FITEE.1601321>.
- [4] Thomas E. Carroll, Michael Crouse, Errin W. Fulp, and Kenneth S. Berenhaut. “Analysis of network address shuffling as a moving target defense”. In: *2014 IEEE International Conference on Communications (ICC)*. June 2014, pp. 701–706. DOI: [10.1109/ICC.2014.6883401](https://doi.org/10.1109/ICC.2014.6883401).
- [5] Yiyang Chang, Ashkan Rezaei, Balajee Vamanan, Jahangir Hasan, Sanjay Rao, and TN Vijaykumar. “Hydra: leveraging functional slicing for efficient distributed SDN controllers”. In: *Communication Systems and Networks (COMSNETS), 2017 9th International Conference on*. IEEE. 2017, pp. 251–258.

-
- [6] Li Da Xu, Wu He, and Shancang Li. "Internet of things in industries: A survey". In: *IEEE Transactions on industrial informatics* 10.4 (2014), pp. 2233–2243.
 - [7] Leslie Daigle. *WHOIS Protocol Specification*. RFC 3912. Sept. 2004. DOI: [10.17487/RFC3912](https://doi.org/10.17487/RFC3912). URL: <https://rfc-editor.org/rfc/rfc3912.txt>.
 - [8] *Developer Survey Results*. Accessed on 27/11/2018. Stack Overflow. 2018. URL: <https://insights.stackoverflow.com/survey/2018/>.
 - [9] Matthew Dunlop, Stephen Groat, William Urbanski, Randy Marchany, and Joseph Tront. "MT6D: A moving target IPv6 defense". In: (Nov. 2011), pp. 1321–1326.
 - [10] Matthew Dunlop, Stephen Groat, William Urbanski, Randy Marchany, and Joseph Tront. "The Blind Man's Bluff Approach to Security Using IPv6". In: *IEEE Security Privacy* 10.4 (July 2012), pp. 35–43. ISSN: 1540-7993. DOI: [10.1109/MSP.2012.28](https://doi.org/10.1109/MSP.2012.28).
 - [11] *Floodlight*. Accessed on 06/7/2018. Project Floodlight. URL: <http://www.projectfloodlight.org/>.
 - [12] Wireshark Foundation. *Wireshark Software, Version 2.4.2*. 2017. URL: <https://www.wireshark.org/>.
 - [13] *GitHut*. Accessed on 27/11/2018. URL: <https://github.info/>.
 - [14] Marc Green, Douglas C. MacFarland, Doran R. Smestad, and Craig A. Shue. "Characterizing Network-Based Moving Target Defenses". In: *Proceedings of the Second ACM Workshop on Moving Target Defense*. MTD '15. Denver, Colorado, USA: ACM, 2015, pp. 31–35. ISBN: 978-1-4503-3823-3. DOI: [10.1145/2808475.2808484](https://doi.org/10.1145/2808475.2808484). URL: <http://doi.acm.org/10.1145/2808475.2808484>.

-
- [15] Stephen Groat, Matthew Dunlop, William Urbanksi, Randy Marchany, and Joseph Tront. "Using an IPv6 moving target defense to protect the Smart Grid". In: *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*. Jan. 2012, pp. 1–7. DOI: [10.1109/ISGT.2012.6175633](https://doi.org/10.1109/ISGT.2012.6175633).
- [16] Vahid Heydari, Sun-il Kim, and Seong-Moo Yoo. "Scalable Anti-Censorship Framework Using Moving Target Defense for Web Servers". In: *IEEE Transactions on Information Forensics and Security* 12.5 (May 2017), pp. 1113–1124. ISSN: 1556-6013. DOI: [10.1109/TIFS.2016.2647218](https://doi.org/10.1109/TIFS.2016.2647218).
- [17] Vahid Heydari, Seong-Moo Yoo, and Sun-il Kim. "Secure VPN Using Mobile IPv6 Based Moving Target Defense". In: *2016 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2016, pp. 1–6. DOI: [10.1109/GLOCOM.2016.7842255](https://doi.org/10.1109/GLOCOM.2016.7842255).
- [18] Jin B. Hong, Seunghyun Yoon, Hyuk Lim, and Dong Seong Kim. "Optimal Network Reconfiguration for Software Defined Networks Using Shuffle-Based Online MTD". In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. Sept. 2017, pp. 234–243. DOI: [10.1109/SRDS.2017.32](https://doi.org/10.1109/SRDS.2017.32).
- [19] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. *Lua 5.1 Reference Manual*. Lua.Org, 2006. ISBN: 8590379833.
- [20] Christopher Jackson. "flexsurv: A Platform for Parametric Survival Modeling in R". In: *Journal of Statistical Software* 70.8 (2016), pp. 1–33. DOI: [10.18637/jss.v070.i08](https://doi.org/10.18637/jss.v070.i08).
- [21] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. "Adversary-aware IP address randomization for proactive agility against sophisticated attackers". In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. Apr. 2015, pp. 738–746. DOI: [10.1109/INFOCOM.2015.7218443](https://doi.org/10.1109/INFOCOM.2015.7218443).

- [22] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. "An effective address mutation approach for disrupting reconnaissance attacks". In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2562–2577.
- [23] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. "Openflow random host mutation: transparent moving target defense using software defined networking". In: *Proceedings of the first workshop on Hot topics in software defined networks*. ACM. 2012, pp. 127–132.
- [24] Keith Jarvis and SecureWorks Counter Threat UnitTM Threat Intelligence. *CryptoLocker Ransomware*. Accessed: 06/01/2019. Secureworks. Dec. 2013. URL: <https://www.secureworks.com/research/cryptolocker-ransomware>.
- [25] John D. Kalbfleisch and Ross L. Prentice. *The Statistical Analysis of Failure Time Data*. JOHN WILEY & SONS INC, Sept. 24, 2002. 462 pp. ISBN: 978-0-471-36357-6. DOI: [10.1002/9781118032985](https://doi.org/10.1002/9781118032985).
- [26] Bob Lantz, Brandon Heller, and Nick McKeown. "A Network in a Laptop: Rapid Prototyping for Software-defined Networks". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. Hotnets-IX. Monterey, California: ACM, 2010, 19:1–19:6. ISBN: 978-1-4503-0409-2. DOI: [10.1145/1868447.1868466](https://doi.org/10.1145/1868447.1868466). URL: <http://doi.acm.org/10.1145/1868447.1868466>.
- [27] Canonical Ltd. *Ubuntu operating system, Version 17.10*. London, United Kingdom, 2018. URL: <https://www.ubuntu.com/>.
- [28] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009. ISBN: 9780979958717.

-
- [29] Douglas C. MacFarland and Craig A. Shue. “The SDN Shuffle: Creating a Moving-Target Defense Using Host-based Software-Defined Networking”. In: *Proceedings of the Second ACM Workshop on Moving Target Defense*. MTD ’15. Denver, Colorado, USA: ACM, 2015, pp. 37–41. ISBN: 978-1-4503-3823-3. DOI: [10.1145/2808475.2808485](https://doi.org/10.1145/2808475.2808485). URL: <http://doi.acm.org/10.1145/2808475.2808485>.
- [30] Hamed Okhravi, Thomas Hobson, David Bigelow, and William Streilein. “Finding Focus in the Blur of Moving-Target Techniques”. In: *IEEE Security Privacy* 12.2 (Mar. 2014), pp. 16–26. ISSN: 1540-7993. DOI: [10.1109/MSP.2013.137](https://doi.org/10.1109/MSP.2013.137).
- [31] Hamed Okhravi, MA Rabe, TJ Mayberry, WG Leonard, TR Hobson, David Bigelow, and WW Streilein. *Survey of cyber moving target techniques*. Tech. rep. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2013.
- [32] *OpenFlow Switch Specification version 1.2*. ONF TS-003. Open Networking Foundation. Dec. 2011.
- [33] Oracle. *VM VirtualBox Software, Version 5.2.12*. 2018. URL: <https://www.virtualbox.org/>.
- [34] *OSI model*. Accessed on 23/7/2018. URL: <https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model>.
- [35] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2018. URL: <https://www.R-project.org>.
- [36] Guido Rossum. *Python Reference Manual*. Tech. rep. Amsterdam, The Netherlands, The Netherlands, 1995.

- [37] Ryu project team. *Ryu SDN Framework*. Ryu SDN Framework Community. 2018. URL: <https://osrg.github.io/ryu-book/en/Ryubook.pdf>.
- [38] Joseph Sagisi, Joseph Tront, and Randy M. Bradley. "Platform agnostic, scalable, and unobtrusive FPGA network processor design of moving target defense over IPv6 (MT6D) over IEEE 802.3 Ethernet". In: *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. May 2017, pp. 165–165. DOI: [10.1109/HST.2017.7951829](https://doi.org/10.1109/HST.2017.7951829).
- [39] Sandra Scott-Hayward. "Trailing the Snail: SDN Controller Security Evolution". In: *arXiv preprint arXiv:1711.08406* (2017).
- [40] Sandra Scott-Hayward, Gemma O'Callaghan, and Sakir Sezer. "Sdn Security: A Survey". In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. Nov. 2013, pp. 1–7. DOI: [10.1109/SDN4FNS.2013.6702553](https://doi.org/10.1109/SDN4FNS.2013.6702553).
- [41] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. "Random host mutation for moving target defense". In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2012, pp. 310–327.
- [42] Dilli Sharma, Dong Seong Kim, Seungyun Yoon, Hyuk Lim, Jin-Hee Cho, and Terrence Moore. "Flexible Random Virtual IP Multiplexing in Software Defined Networking". In: *TrustCom* (2018).
- [43] Yuan Shi, Huanguo Zhang, Juan Wang, Feng Xiao, Jianwei Huang, Daochen Zha, Hongxin Hu, Fei Yan, and Bo Zhao. "CHAOS: An SDN-Based Moving Target Defense System". In: *Security and Communication Networks 2017* (2017), 3659167:1–3659167:11.
- [44] *Simple HTTP Server*. Accessed on 05/11/2018. Python Software Foundation. URL: <https://docs.python.org/2/library/simplehttpserver.html>.

-
- [45] William Stallings and Lawrie Brown. *Computer Security: Principles and Practice*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2014. ISBN: 0133773922, 9780133773927.
- [46] Symantec. "Internet Security Threat Report". In: 20 (2015).
- [47] Laszlo Szekeres, Mathias Payer, Tao Wei, and Dawn Song. "SoK: Eternal War in Memory". In: *2013 IEEE Symposium on Security and Privacy*. May 2013, pp. 48–62. DOI: [10.1109/SP.2013.13](https://doi.org/10.1109/SP.2013.13).
- [48] Terry M. Therneau. *A Package for Survival Analysis in S*. version 2.38. 2015. URL: <https://CRAN.R-project.org/package=survival>.
- [49] Li Wang and Dinghao Wu. "Moving target defense against network reconnaissance with software defined networking". In: *International Conference on Information Security*. Springer. 2016, pp. 203–217.
- [50] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: <http://ggplot2.org>.
- [51] Jun Xu, Pinyao Guo, Mingyi Zhao, Robert F. Erbacher, Minghui Zhu, and Peng Liu. "Comparing Different Moving Target Defense Techniques". In: *Proceedings of the First ACM Workshop on Moving Target Defense*. MTD '14. Scottsdale, Arizona, USA: ACM, 2014, pp. 97–107. ISBN: 978-1-4503-3150-0. DOI: [10.1145/2663474.2663486](https://doi.org/10.1145/2663474.2663486). URL: <http://doi.acm.org/10.1145/2663474.2663486>.
- [52] Saman Taghavi Zargar, James Joshi, and David Tipper. "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks". In: *IEEE communications surveys & tutorials* 15.4 (2013), pp. 2046–2069.

- [53] Kimberly Zeitz, Michael Cantrell, Randy Marchany, and Joseph Tront. “Changing the Game: A Micro Moving Target IPv6 Defense for the Internet of Things”. In: *IEEE Wireless Communications Letters* (2018), pp. 1–1. ISSN: 2162-2337. DOI: [10.1109/LWC.2018.2797916](https://doi.org/10.1109/LWC.2018.2797916).
- [54] Kimberly Zeitz, Michael Cantrell, Randy Marchany, and Joseph Tront. “Designing a Micro-moving Target IPv6 Defense for the Internet of Things”. In: *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*. Apr. 2017, pp. 179–184.
- [55] Rui Zhuang, Su Zhang, Alex Bardas, Scott A. DeLoach, Xinming Ou, and Anoop Singhal. “Investigating the application of moving target defenses to network security”. In: *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*. IEEE. 2013, pp. 162–169.